Geospatial Computing: Architectures and Algorithms for Mapping Applications

Richard William Milton

Submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy

The Bartlett Centre for Advanced Spatial Analysis
University College London

16 February 2019

I, Richard William Milton, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

. . .

Abstract

Beginning with the MapTube website,[1] which was launched in 2007 for crowd-sourcing maps, this project investigates approaches to exploratory Geographic Information Systems (GIS) using web-based mapping, or 'web GIS'. Users can log in to upload their own maps and overlay different layers of GIS data sets. This work looks into the theory behind how web-based mapping systems function and whether their performance can be modelled and predicted.

One of the important questions when dealing with different geospatial data sets is how they relate to one another. Internet data stores provide another source of information, which can be exploited if more generic geospatial data mining techniques are developed. The identification of similarities between thousands of maps is a GIS technique that can give structure to the overall fabric of the data, once the problems of scalability and comparisons between different geographies are solved. After running MapTube for nine years to crowd-source data, this would mark a natural progression from visualisation of individual maps to wider questions about what additional knowledge can be discovered from the data collected.

In the new 'data science' age, the introduction of real-time data sets introduces a new challenge for web-based mapping applications. The mapping of real-time geospatial systems is technically challenging, but has the potential to show inter-dependencies as they emerge in the time series. Combined geospatial and temporal data mining of real-time sources can provide archives of transport and environmental data from which to accurately model the systems under investigation. By using techniques from machine learning, the models can be built directly from the real-time data stream. These models can then be used for analysis and experimentation, being derived directly from city data. This then leads to an analysis of the behaviours of the interacting systems.

[1] The MapTube website: *http://www.maptube.org.*

Impact Statement

This thesis examines the performance of web-based and real-time mapping systems, developing the theory behind how the systems work at a technical level, while quantifying the performance using results from testing in the field. One of the impacts of this work is to provide researchers with a basis from which to design and build their own geographic visualisation systems.

As the work covers the period from 2005 to 2018, the main impact has been in the projects that this research has been used to deliver. The graph in figure 4.3 of Chapter 4 shows that CASA's original GMapCreator software had 20,928 downloads up to January 2014. This was the first impact from the work which then turned into the contents of this thesis. Following on from this, the MapTube website, presented in Chapter 4 for crowd-sourcing maps online, has 16,898 registered users and 3,446 maps. Given the way that MapTube collects data, a significant part of this thesis deals with how to perform geospatial analysis on sets of maps as a new method of working with Internet data stores containing information.

The crowd-sourcing work is taken a stage further with the 'mood maps' in section 4.1, which was a project where CASA collaborated with the BBC. The first experiment had 23,475 responses to a question about the impact of the credit crunch, with a follow-up a few months later having 20,072 responses. Then, Manchester City Council ran a survey with us in conjunction with the public consultation surrounding the introduction of a congestion charging zone, obtaining 15,800 responses from the general public. This led to further funding through ESRC for a dedicated website for crowd-sourcing geographic information called 'SurveyMapper'.

In terms of the core mapping technologies and theory presented here, the latest project developed the QUANT website, which required the vector tiling technology (section 5.1.4). This was developed out of the need to create a geographic data exploration tool to visualise the output of a spatial interaction model. Continuing the pace

of technological development, this has led to a successful commercial bid to supply a map-based visualisation of building energy data for the Greater London Authority in conjunction with UCL's Energy Institute.

Where the real-time data chapter is concerned, this has enabled a collection of outreach projects involving visualisation, for example, Steven Gray's iPad wall, which CASA installed in the Mayor's office in City Hall, the 'Grand Designs' exhibition at the ExCel Centre in London, a three day exhibition at Leeds City Museum and numerous other 'Smart City' projects. It currently provides real-time data to the CityDashboard website and is also used for another commercial project with the London Transport museum, as detailed in the Conclusion Chapter. In addition to this, the data has been collected since the London Olympics of 2012, so there now exists an archive of 6 years' worth of data for researchers to analyse. The twenty publications resulting from this work are listed in the following Preface Chapter.

Contents

List of Tables

List of Figures

Preface

The central theme running through this thesis is developing new tools and methodologies for the extraction of knowledge from geospatial data. In the course of this work, many different types of geospatial data are explored, including environmental data, weather data, transport data and archives of data in data stores. Starting with the weather and environmental data in chapters 5.1.3 and 8.4, this work was influenced by the earliest papers I published. Beginning in 2004 with the EQUATOR eScience project[2], my work for the paper, "Data Visualization within Urban Models" [Ste+04], involved extending a 3D city modelling system with data from GPS tracked carbon monoxide sensors. Spanning approximately 18 months, this work led to the journal article, "Mapping Carbon Monoxide Using GPS Tracked Sensors" [MS07], in which I published an analysis of the data I had collected around the Clerkenwell area of London. An updated paper, "Using Tracked Mobile Sensors to Make Maps of Environmental Effects", was published in the journal, "Personal and Ubiquitous Computing" [SM08] in 2008 and the work was the subject of an article in New Scientist [Rei06], being one of the first experiments with mobile, tracked, environmental sensors. In addition to this, "Correcting GPS Readings from a Tracked Mobile Sensor" published in "Location and Context Awareness" [MS05] was a paper specifically focused on the correction methodologies that I used with the GPS tracks. All these publications form the basis of the environmental data analysis in section 8.4 of this thesis where I use the technique of clustering according to weather conditions from [MS07] to analyse air quality data in the context of a year's worth of public transport data.

Moving to the Centre for Advanced Spatial Analysis (CASA) in 2005, the work on 3D visualisation continued with the Geographic Virtual Urban Environments (GeoVUE) project[3]. The first GeoVUE paper which I contributed to was "Scaling and allometry

[2]The EQUATOR work was supported by the UK projects Advanced Grid Interfaces for Environmental e-science in the Lab and in the Field (EPSRC Grant GR/R81985/01) and EQUATOR Interdisciplinary Research Collaboration (EPSRC Grant GR/N15986/01).

[3]The Geographic Virtual Urban Environments (GeoVUE) work was supported by ESRC Grant RES-149-25-1023.

in the building geometries of Greater London" [Bat+08], with my work being to generate the geometric statistics on building perimeter, area, height and volume for the dataset containing all of London's 3.6 million buildings. The rank size relationships link with my earlier work on pollution, where the street canyon ratio of width to height is a contributing factor for pollution levels. These ideas of using the geometry of the city as a part of the analysis are mentioned again later in this thesis in section 8.4 on environmental data.

Geometric structure aside, the bulk of the GeoVUE work delivered the first release of my MapTube website, which is introduced in the next section and forms the main core of the work in this thesis. Initially conceived as a crowd-sourcing site for geospatial data, the first MapTube publications, "Mapping for the Masses, Accessing Web 2.0 Through Crowdsourcing" [Hud+09b], "NeoGeography and Web 2.0: Concepts, Tools and Applications" [Hud+09a] and the "Origins of MapTube" article in Civil Engineering Surveyor magazine (June 2009), focus on my 'GMapCreator' software which was used to create the first maps on MapTube. In these three publications and in the later publication, "Map Mashups, Web 2.0 and the GIS Revolution" [Bat+10b], the concept of a 'Mood Map' is introduced. These were online spatial surveys, starting with a collaboration between CASA and the BBC's Radio 4 and Newsnight programmes to run a survey asking about the general public's opinion to the current financial situation, which the media referred to as the 'Credit Crunch'. This initial survey received 23,000 responses in total and was followed by a number of others, most notably one about the proposed Manchester city centre congestion charge. This was a collaboration between the National Centre for e-Social Science (NCeSS), Manchester City Council, BBC North and CASA, with the publicity from the BBC resulting in 15,902 responses to the survey. This work was carried out as part of the Generative eSocial Science Project (GenESiS)[4] and JISC National e-Infrastructure for Social Simulation (NeISS) project. In "Calibration of a Spatial Simulation Model with Volunteered Geographic Information" [Bir+11] and "Elements of a Computational Infrastructure for Social Simulation" [Bir+10], my data from the Manchester congestion charge survey is used to calibrate a land use transport model with MapTube used for the visualisation. This is a new concept, linking volunteered geographic information to the calibration of an urban model. In this publication and in "Map Mashups, Web 2.0 and the GIS

[4]The Generative E-Social Science (GenESiS) project was supported by ESRC grant RES-149-25-1078.

Revolution" [Bat+10b], the link is made between urban modelling and web based mapping, turning visualisation of static geographic data into exploratory visualisation of model outputs. This concept of viewing the web mapping system as analogous to the visualisation layer in a conventional GIS is integral to the dynamic mapping approach proposed in this thesis, with the MapTube 'vector tiler' of section 5.1.4 a principle component. The NeISS system architecture in figure 1 of [Bir+11] shows the MapTube system used as a web service at the end of a modelling chain. The later GenESiS work delivered a dynamic version of MapTube, as a web service able to make dynamic maps from data on demand in a matter of seconds. This is the critical infrastructure element with MapTube used as the mapping component in a more general tool chain. In the process of building this infrastructure, other avenues were also explored, such as exporting MapTube maps as 3D content into Linden Labs' 3D online virtual environment, called 'Second Life'. In "Virtual Cities: Digital Mirrors into a Recursive World" [Hud+07b], maps from MapTube and small parts of Virtual London are shown on Nature Magazine's 'Second Nature' Island. This link between the two dimensional and three dimensional, or immersive worlds, is also covered in the JISC Technology and Standards Watch document (TechWatch), "Data Mash-ups and the Future of Mapping" [Bat+10a]. Here I review the Open Geospatial Consortium, or OGC's, current standards in relation to the emerging ad-hoc standards of the web-based tiled maps. The crowd-sourced surveys of MapTube are also covered, providing a contrast between the OGC's Web Mapping Standard (WMS) and the new tiled maps which are designed to handle 'data at scale'. This forms the central argument of chapter 4 where I show how the MapTube architecture evolved between 2008 and 2011 in order to handle surveys where television publicity meant receiving a million responses in an hour (figure 4.31 in chapter 4). The book chapter, "Advances in Crowdsourcing" [GMH15], rounds off the crowdsourcing work with my analysis of how people had tried to manipulate the Manchester congestion charge survey.

Continuing the trend towards making it easier to make maps and so collect more geospatial data, the JISC National e-Infrastructure for Social Simulation (NeISS) project extended the MapTube technology in a novel and unexpected way. The work involved uploading data from the 2001 Census, necessitating the creation of 145 new maps[5], which, in turn, required greater automation in the mapping process. Ultimately leading

[5] See: *http://digitalurban.org/2009/10/maptube-now-with-145-census-maps.html*.

to the idea of "Data Store Mining", which forms a central thread of this thesis, section 4.4.1 uses concepts from data mining to automatically make maps.

Following on from the GenESiS grant was the ESRC 'Talisman' project[6], a research node under the National Centre for Research Methods. The Talisman project turned the method used for bulk uploading Census data into an automatic data mapping system. The work in section 5.2 shows that this technique, when applied to an Internet data store, can show relationships between datasets and give structure to the information, answering the question, "What does a data store look like?". Text domain correlation (section 5.2.1) and pattern matching (section 5.2.2) complete the tool kit of methods, culminating in the final analysis of all the 2011 Census data in section 7.1, followed by an open source project to compare data based on different geographies.

The automatic mapping work was first published in the book chapter, "Visualizing Spatial and Social Media" [Bat+15], along with the work from another project on real-time transport data. This project was called "Adaptive Networks for complex Transport Systems", or "ANTS" for short, and was a 2 month project funded by Future ICT's "Big Data in the City" call. This delivered real-time data about the London Underground system which could be included in MapTube, forming the core of chapter 6. My work on the project is to appear as a book chapter in the forthcoming, "CyberGIS: Fostering a New Wave of Geospatial Discovery. CyberGIS for Analysing Urban Data", [Che+14] and [MGH13]. A spin-off project was the CityDashboard website [GMH13], which consumes the real-time data generated by the ANTS project. In addition to this, real-time data about London has been archived since July 2012, providing the archive of data which the analysis in chapters 6 and 8 depend on. Finally, in the book chapter, "Smart London" [Bat+13] and also in "Finding Pearls in London's Oysters" [Rea+16], this new source of automatically generated data about London is analysed in detail.

The next chapter outlines the development of the MapTube website, demonstrates the automatic map making technology and finishes with real-time data about London's transport systems.

Finally, due to the number of acronyms and technical terms used in this document, a glossary is included in the appendices at the end for reference.

[6]The Talisman project, Geospatial Data Analysis and Simulation, was supported by ESRC grant RES-576-25-0039.

List of Publications

The following lists all my publications from 2004 to the present day.

2019

Accelerating Urban Modelling Algorithms with Artificial Intelligence. **MILTON, R.**, and Roumpani, F. *5th International Conference on Geographical Information Systems Theory, Applications and Management*, May 3-5, 2019, Heraklion, Crete Greece, (accepted).

Internet of Things of Trees - Conversational Objects via SMS Protocols. de Jode, M., Lovett, L., Hay, D., Hudson-Smith, A., **MILTON, R.**, and Fraser, L. *PETRAS Living in the Internet of Things*, Institute of Engineering Technology Trustworthy IoT, May 2019, (accepted).

2018

Talking to GNOMEs: Exploring Privacy and Trust Around Internet of Things Devices in a Public Space. **MILTON, R.**, Buyuklieva, B., Hay, D., Hudson-Smith, A. and Gray, S., *CHI'18 Extended Abstracts*, April 21–26, 2018, Montreal, QC, Canada, ISBN 978-1-4503-5621-3/18/04, *https://doi.org/10.1145/3170427.3188481*.

Smart IoT and Soft AI. **MILTON, R.**, Hay, D., Gray, S., Buyuklieva, B. and Hudson-Smith, A., *PETRAS Living in the Internet of Things*, Institute of Engineering Technology Trustworthy IoT, October 2018.

IoT in the Wild: what negotiating public deployments can tell us about the state of the Internet of Things. Hay, D., Buyuklieva, B., Daothong, J., Edmonds, B., Hudson-Smith, A., **MILTON, R.**, Wood, J., *PETRAS Living in the Internet of Things*, Institute of Engineering Technology Trustworthy IoT, October 2018.

2016

Finding Pearls in London's Oysters. Reades, J., Zhong, C., Manley, E., **MILTON, R.** and Batty, M., Built Environment Volume 42, Number 3, Special Issue: Big Data and the City, October 2016, 365-381

2015

Visualizing Spatial and Social Media. Halfpenny, P. (Editor) and Proctor, R. (Editor), Batty, M., Gray, S., Hudson-Smith, A., **MILTON, R.**, O'Brien, O. and Roumpani, F., Innovations in Digital Research Methods. SAGE Publications Ltd., (ASIN B012HU8OAS), 1 June 2015, 245-270

Advances in Crowdsourcing: Surveys, Social Media and Geospatial Analysis: Towards a Big Data Toolkit. Gray, S., **MILTON, R.**, Hudson-Smith, A., Advances in Crowdsourcing. Springer International Publishing, DOI 10.1007/978-3-319-18341-1_13, F.J. Garrigos-Simon et al. (eds.).

CyberGIS for Analyzing Urban Data. Cheshire, J., Batty, M., Reades, J., Longley, P., Manley, E. and **MILTON, R.** (pending). Wang, S. and Goodchild, M. (eds). CyberGIS: Fostering a New Wave of Discovery and Innovation. Springer-Verlag.

2013

Visualising Spatial and Social Media, CASA Working Paper 190. Batty, M., Gray, S., Hudson-Smith, A., **MILTON, R.**, O'Brien, O. and Roumpani, F. (2013). ISSN 1467-1298.

Visualizing real-time data with an interactive video wall. Gray, S., **MILTON, R.**, Hudson-Smith, A. (2013). NCRM Methods News Spring 2013.

2011

Calibration of a spatial simulation model with volunteered geographical information. Birkin, M., Malleson, N., Hudson-Smith, A., Gray, S., **MILTON, R.** (2011). International Journal of Geographical Information Science 25(8) 1221-1239.

2010

Data mash-ups and the future of mapping. **MILTON, R.**, Anand, S., Batty, M., Crooks, A., Hudson-Smith, A., Jackson, M., and Morley, J. (2010). (JISC Techwatch, pp. 1-46). UK: JISC.

2009

MapTube Origins. **MILTON, R.** (2009). Civil Engineering Surveyor, June 2009, 30-32.

Neogeography and Web 2.0: Concepts, Tools and Applications. Hudson-Smith, A., Crooks, A., Gibin, M., **MILTON, R.**, and Batty, M. (2009). Journal of Location Based Services, 3 (2), June 2009, 118-145.

The Neogeography of virtual cities: digital mirrors into a recursive world in Handbook of Research on Urban Informatics. Batty, M., Hudson-Smith, A., **MILTON, R.**, and Dearden, J. (2009). M. Foth (Ed.), IGI Global Snippet.

Crowdsourcing Spatial Surveys and Mapping. Crooks, A. T., Hudson-Smith, A. M., **Milton, R.** and Batty, M. (2009). Fairbairn, D. (ed.), Proceedings of the 17th Geographical Information Systems Research UK Conference, Durham University, England.

2008

Scaling and allometry in the building geometries of Greater London. Batty, M., Carvalho, R., Hudson-Smith, A., **MILTON, R.**, Smith, D. and Steadman, P. (2008). In: The European Physical Journal B 63 (2008), pp. 303314. DOI: 10.1140/epjb/e2008-00251-5.

Using tracked mobile sensors to make maps of environmental effects. Steed, A., and **MILTON, R.** (2008). Personal and Ubiquitous Computing, 12 (4), 331-342.

2007

Mapping Carbon Monoxide using GPS Tracked Sensors. **MILTON, R.**, Steed,A. (2007). Environmental Monitoring and Assessment. 124(1-3), 1-19. ISSN: 01676369.

2005

Correcting GPS Readings from a Tracked Mobile Sensor. **MILTON, R.**, Steed, A. (2005). Location and Context Awareness 2005, Strang, T., Linnho-Popien, C. (ed.) Lecture Notes in Computer Science series. , 3479, 83-94.

2004

Data Visualisation within Urban Models. Steed, A., Spinello, S., Croxford, B., **MILTON,R.** (2004). Theory and Practice of Computer Graphics, Jones, M. W. (ed.) IEEE Computer Society, 9-16.

Software

The following list contains all the software I have written which is relevant to this thesis. Where code is open source, the repository is listed under '*Code*', otherwise the intellectual property rights belong to the project and not myself. Where the project provides an open resource available to the general public, either the website, download link or web service is listed in place of the code section. Finally, the '*Referenced in*' field provides a cross reference to the chapter and section number in this document where the code is referred to, with the bold entry indicating the primary location.

GMapCreator

Date:	2006.
Program:	17,042 lines, Java.
Project:	Geographic Virtual Urban Environments (GeoVUE), ESRC RES-149-25-1023.
Purpose:	Make a web page containing a Google Map from data in an ESRI shapefile.
Referenced in:	4.1, 4.2, **4.3**
Publications:	[Mil09], [Hud+09a], [Hud+09b], [Bat+10a], [Bat+10b], [Bir+11].

MapTube

Date:	20 February 2008.
Website:	*http://www.maptube.org*.
Program:	54,061 lines (core), 3,378 (API), 4,101 (js), 7,678 (html), C#, ASP.net, HTML, Javascript.
Project:	Geographic Virtual Urban Environments (GeoVUE), ESRC RES-149-25-1023.
Purpose:	Website for the online sharing of maps using Google Maps or OpenStreetMap as base layers. The aim is to make it easier for people to upload geospatial information and so collect more data.
Referenced in:	4.1
Publications:	[Mil09], [Hud+09a], [Hud+09b], [Bat+10a], [Bat+10b], [Bir+11], [Bat+15], [GMH15].

MapTubeD

Date:	2008.
Web Service:	*http://shoal.casa.ucl.ac.uk/TileRequestHandler.ashx*.
Program:	13,363 lines, C#.
Project:	Generative E-Social Science (GenESiS) ESRC RES-149-25-1078.
Purpose:	Renders PNG image tiles for Google Maps from Internet based data. Its function is a render on-demand tile renderer for MapTube. Essential component in *http://surveymapper.org*.
Referenced in:	4.1, **4.4**
Publications:	[Bir+11], [Bat+15].

GeometryFinder

Date:	1 October 2008 to 30 September 2012
Web Service:	*http://shoal.casa.ucl.ac.uk/GeometryFinderHandler.ashx*.
Program:	911 lines, C#.
Project:	Generative E-Social Science (GenESiS) ESRC RES-149-25-1078.
Purpose:	Automatically detect the spatial context of data contained in a CSV or shapefile and make a map from the data automatically. Used by MapTube and relies on MapTubeD for the tile rendering.
Referenced in:	5.2
Publications:	None.

MapTubeV

Date:	2016
Code:	*https://github.com/maptube/MapTubeV*.
Program:	1,743 lines, C#.
Project:	PhD.
Purpose:	Variant of MapTubeD, except that it returns MapBox protocol buffer format tiles as vectors.
Referenced in:	5.1.4
Publications:	None, although it is the technology behind the QUANT website: *http://quant.casa.ucl.ac.uk*.

Data Store Miner

Date:	2010
Code:	*https://github.com/maptube/DataStoreMiner*.
Program:	2,052 lines, C#.
Project:	Talisman, Geospatial Data Analysis and Simulation ESRC RES-576-25-0039.
Purpose:	Core library which provides generic functions for writing code to extract and analyse data from data stores or other sources of bulk download (e.g. the NOMIS Census bulk release). This relies heavily on the Geometry Finder project and MapTubeD.
Referenced in:	4.4.1, 4.4.2, **5.2**, 7.1
Publications:	None.

GeoGL

Date:	2011.
Code:	*https://github.com/maptube/GeoGL.*
Program:	17,328 lines, C++.
Project:	PhD.
Purpose:	3D visualisation and agent based simulation. Used to generate statistics from a year's worth of transport and environmental data collected by *CityDB API*.
Referenced in:	6.3
Publications:	None.

MapTubeExplorer

Date:	2016.
Code:	*https://github.com/maptube/MapTubeExplorer*.
Program:	213 lines, C#.
Project:	PhD.
Purpose:	Performs spatial correlation between a test map and all the maps currently stored on MapTube.
Referenced in:	7.1.3
Publications:	None.

Meteorological Surface Observation Decoder

Date:	2003.
Code:	*https://github.com/maptube/WxDecoder*.
Program:	3,111 lines, C++.
Project:	Private.
Purpose:	Parsing of coded meteorological bulletins into CSV files. Used for the environmental data exploration section.
Referenced in:	4.5, **5.1.3**, 8.4
Publications:	[MS07], [GMH15, for the snow maps].

Adaptive Networks for complex Transport Systems (ANTS)

Date:	2012.
Program:	27,931 lines, C#.
Project:	ANTS (Future ICT).
Purpose:	Real-time tracking of 450 London Underground tubes, 900 Network Rail trains, 7,000 TfL buses, 12 TfL river boats, 9,600 bikes and 17,960 traffic sensors[7].
Referenced in:	4.5, **6.2**
Publications:	[MGH13], [Bat+13], [Che+14].

CityDB API

Date:	2012.
Web Service:	*http://loggerhead.casa.ucl.ac.uk/api/s/trackernet/json*.
Program:	6,613 lines, C#.
Project:	Talisman, Geospatial Data Analysis and Simulation ESRC RES-576-25-0039.
Purpose:	To collect data from real-time sources and provide an end-point for querying the data. Uses the ANTS library for transport data processing, adding environmental and weather data. Used by *citydashboard.org* for transport data and by the 'iPad wall project' in [Bat+15].
Referenced in:	4.1, **4.5**
Publications:	[MGH13], [GMH13], [Che+14], [Bat+15].

Chapter 1

Introduction

1.1 Motivation and Context for the Work

The context for this work draws on experience with developing web-based mapping systems from when Google Maps was first released in February 2005, up until the present day. During this time, technological innovation has seen web-based mapping evolve from simple cartography and static, tiled, images into handling vector geometry with dynamic data attributes. The ultimate evolution is a fully functional "WebGIS" with spatial analytics capabilities, but, first, there are a number of technological challenges to solve. Part of this thesis explores the fundamental architectures and algorithms that are required to make this a reality, while the more applied parts investigate how this can be used by researchers working on real world problems.

In the process of making maps more accessible to the general public, tools for automatically making maps from data are now an integral part of the data preparation pipeline. These emerging, intelligent, tools are one aspect of this research, along with the "Internet quantities" of data available and infinite provisioning. The ability to handle complex, inter-related, geospatial data at scale using intelligent tools provides the primary motivation for the research presented in this thesis.

Real-time city data from application programming interfaces (APIs) and permanently connected streams are now available for London and other cities, providing information on tubes, buses, trains, bikes, weather and air quality. The move to make cities 'Smart', as defined in IBM's publication, "Smarter Cities Series: A Foundation for Understanding IBM Smarter Cities" [Keh+11], is resulting in data about cities becoming more open. In city simulation, the concept of the 'Digital Twin' originally comes from production engineering, as defined in the article by Grieves, "Origins of the Digital Twin Concept" [Gri16]. He defines the Digital Twin as follows:

> "It is based on the idea that a digital informational construct about a physical system could be created as an entity on its own. The digital information would be a "twin" of the information that was embedded within the physical system itself and be linked with that physical system through the entire lifecycle of the system." (Grieves [Gri16])

This refers to the digital twin in the context of physical products, or devices, highlighting a "predictive" and "interrogative" Digital Twin Environment for acting on digital twins. Predictive is an environment designed to predict future behaviour, for example predictive component failures from specific instances of physical products in the real world, utilising known manufacturing tolerances. Interrogative is closer to the relationship with digital twins of cities, which Grieves describes as follows:

> "Irrespective of where their physical counterpart resided in the world, individual instances could be interrogated for their current system state: fuel amount, throttle settings, geographical location, structure stress, or any other characteristic that was instrumented." (Grieves [Gri16])

The examples given are all engineering related, though, citing "space exploration", "next generation fighter aircraft" and "NASA vehicles". Digital twins of cities are discussed by Batty in his editorial, "Digital twins" [Bat18]. The topic of computer simulation in the context of Digital Twins and Smart Cities is raised, arguing that, "a computer model of a physical system can never be the basis of a digital twin for many elements of the real system are ignored in any such abstraction" [Bat18]. In general, the idea of the digital twin mirroring the original system is a modelling problem, depending on the model builder's choice of key factors to include.

> "Models are, by definition, simplifications of the real thing and in that sense, do not aim to replicate the original system in the same detail as that system." (Batty [Bat18])

Where the idea of a digital twin is referenced in the context of real-time city data, this is not so much a functional twin as it is Grieves' interrogative view of the system. The system here is the unknown element under investigation, with the interrogative view of the "fuel amount and throttle settings" providing the researcher's only view into its internal operation. A city is also a connected system of systems, which resists the

type of decoupling and isolated analysis which would be a first step in the engineering sciences. City systems cannot ordinarily be decoupled unless one element of the system fails, for example during a tube or bus strike. This is where long-term monitoring of the respective systems via data APIs can be useful in building a normal operating point as an aid to formulating a theory about how the system functions.

Central to this view of connected data systems are the fundamental algorithms which make the analysis possible. Computer architecture plays a role here, with parallelism enabling higher throughputs of data for any algorithms capable of exploiting it. Visualisation of the data is a primary aim, so different ways of enabling researchers to see their data in a spatial context are investigated. The quantities of spatial data now available on the Internet and the availability of real-time city data through this medium form the core of this thesis. The term "WebGIS" is used here to describe any form of spatial analysis or spatial visualisation using a web browser, however simplistic this might be. The idea is attractive from the point of view of lowering the bar to entry for non-expert users, while simultaneously simplifying the spatial analysis pipeline for everybody. From a computer architecture and algorithms point of view, this is an application of emerging and fast developing technologies to a real world problem.

The next section develops this into a set of research questions.

1.2 Research Question

Q1. How do automated mapping tools, real-time data and web-based mapping alter the landscape of geospatial analysis in an era of ever-increasing quantities of Internet data?

This can be broken down into the following sub-questions:

Q1.1 WebGIS

Q1.1 What computer architecture satisfies the "WebGIS" requirement, taking into account storage and performance?

Q1.2 Map Comparisons

Q1.2 The ability to build maps automatically from data and populate a geospatial data store raises questions about how connections and similarities between maps can be formed, leading to the discovery of new knowledge about the structure of these relationships.

Q1.3 Real-time GIS and Agent-based Models

Q1.3 Real-time geospatial data is a geospatial agent-based model evolving over time. Can the rules that make the system work be learnt from the real-time stream of data?

Q1.4 Combined Sources of Data

Q1.4 What links exist between static and real-time data analysis?

1.3 Project Overview Diagram

The diagram in figure 1.1 shows an overview of all the elements contributing to the work in this thesis:

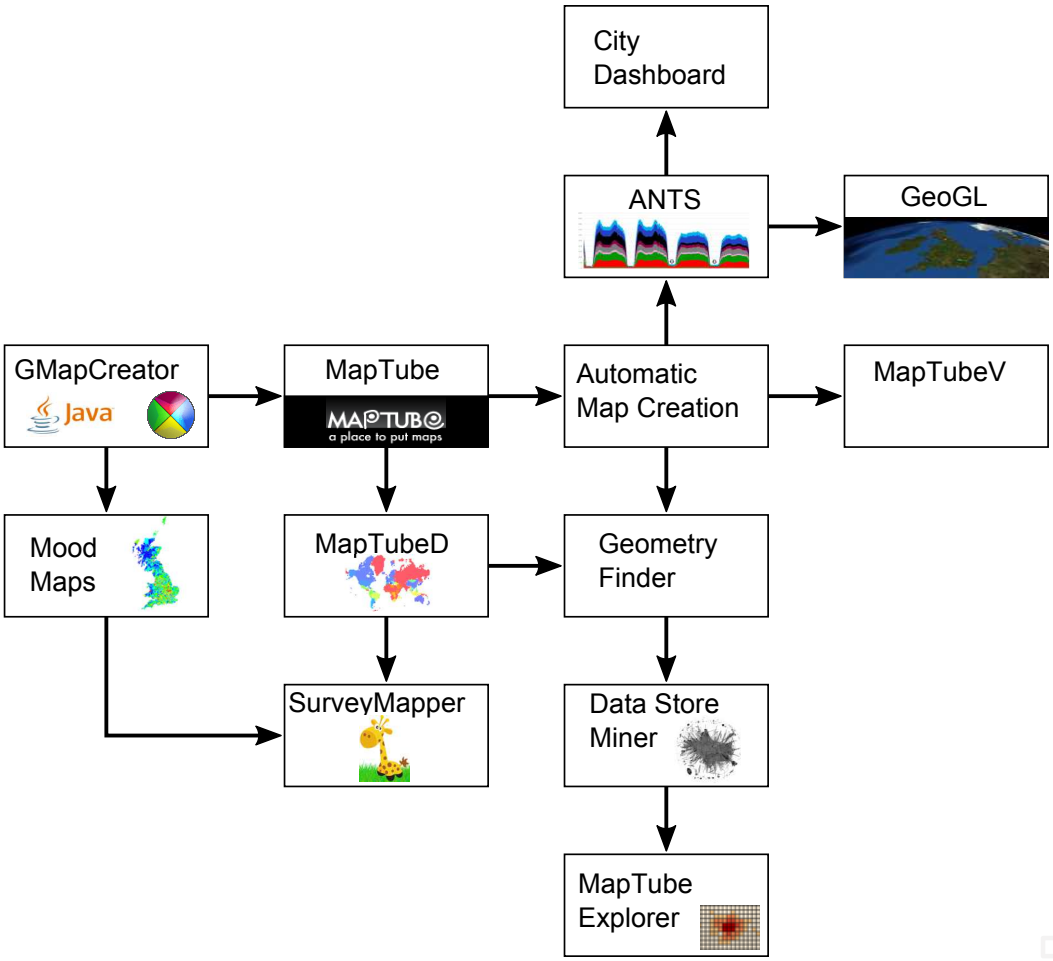


Figure 1.1: Overview of projects contributing to this thesis. These projects link to the descriptions in the "Software" Chapter of the preface.

Starting with the GMapCreator, which was a Java application to create a working Google Maps site from data contained in a shapefile, this is the beginning of the tiled mapping technology. Shortly afterwards, the idea to use this to crowd-source maps was conceived and MapTube was built around the sharing of GMapCreator tile sets. In parallel with this, Prof. Andy Hudson-Smith created the idea of a 'mood map' and negotiated with Radio Four's iPM programme for them to publicise an online question about the 'Credit Crunch' which built a map automatically from their viewers' responses. The technology behind this was the GMapCreator, but it could only rebuild the map once every 30 minutes, so the MapTubeD dynamic tile rendering service filled this gap, providing a spin-off in the NeISS project called Survey Mapper, which was an-

other website built by Steven Gray for user-created online surveys. The MapTubeD tile renderer gave MapTube the ability to build maps automatically from spatial attribute data contained in CSV files with the inclusion of the automatic map generation code. In addition to this, the 'Geometry Finder' part of the MapTubeD code was separated and exposed as a web service with this and the automatic mapping code combining to form the basis for the 'Datastore Miner' project for making maps automatically from data in Internet data stores, leading to the map comparison and 'MapTube Explorer' projects which form a large part of this thesis. Following the other routes from the automatic mapping technology, 'MapTubeV' is a vector tiler which enables greater interactivity with web-based maps, taking a big step towards WebGIS. Real-time data visualisation follows the final path via the 'Adaptive Networks for complex Transport Systems' project which I proposed, providing real-time transport data for another spin-off project called "CityDashboard". Finally, the "GeoGL" project was developed solely for the real-time data processing and visualisation essential for this thesis, providing computationally intensive agent-based modelling and analytics functionality on a virtual globe for quantities of data which could not be processed using the web based MapTube libraries. Everything stated here is part of this thesis, apart from the SurveyMapper and CityDashboard projects which were developed by other researchers on separate grants. MapTube provides the map rendering infrastructure to SurveyMapper, while ANTS supplies bus and tube data to CityDashboard.

1.4 Timeline

As already stated in the preface, the work in this thesis was conducted while working as a researcher on six different academic projects. The Equator project with UCL Computer Science is only mentioned as it contributed the papers I wrote on air quality monitoring. Then, working for the Centre for Advanced Spatial Analysis (CASA), the GeoVUE, GenESiS, NeISS, Talisman, and ANTS FutureICT projects make separate contributions. The web-based mapping work started with GeoVUE, continuing into GenESiS, and provided the foundation to the project proposal that made the JISC NeISS grant successful, delivering the first Census maps project and SurveyMapper. The NCRM Talisman grant was written around geospatial data mining, real-time city data and visualisation, building on what had been previously started with my data store mining project which began as an idea for a PhD topic while working on GenESiS.

Real-time data completes the picture with the FutureICT funded project, ANTS, which I pitched as an idea at a funding meeting and worked on with three other researchers for two months, although all the code development is mine. A time-line of the development is shown in Table 1.1.

Table 1.1: Time-line of Work

2004	Equator project in UCL Computer Science. This is only mentioned for the papers on air quality which are referenced later.
2005	GeoVUE project starts in CASA.
2006	GMapCreator released for automatically building Google Maps websites from data in shapefiles.
2008	MapTube website released on 20th February at the Barbican Centre in London. The idea behind MapTube is to crowd-source geospatial data.
	GenESiS project starts in CASA.
	Mood Maps, volunteered geographic information project created by Prof. Andy Hudson-Smith and initially trialled on Radio Four's iPM programme as a "Credit Crunch" question. This is the genesis of the mapping on demand technology.
	MapTubeD, MapTube Dynamic tile renderer. This was built to make the mood maps render immediately from fast changing data, but also has applications for automatic mapping on MapTube.
	Geometry Finder web service included as a spin-off from MapTubeD. This is a piece of infrastructure which allows automatic identification of the spatial context of data.
2010	Talisman project starts in CASA.
	Data Store Miner, first envisaged following the launch of the London Datastore in the same year. This takes the automatic mapping technology to a new level by mapping all the geospatial data on the website. Map comparisons with Census data follows on from this.
2011	GeoGL, virtual globe project for 3D rendering real-time data. Used for analysis of large amounts of real-time data not possible with the web based systems.
2012	ANTS, Adaptive Networks for complex Transport Systems project starts in CASA (duration 3 months). Provides real-time transport data to other systems.
	CityDBAPI, data downloading and query API for real-time data which is used by other systems. Handles, transport data (via ANTS), cycle hire, weather and air quality data.
2016	MapTubeV, vector tiler for MapTube. Adds the ability to visualise 'one-off' model outputs.
	MapTubeExplorer, provides map comparisons between maps on MapTube where the geometry may not be identical. Built as an experimentation tool for map comparison techniques.

The audience for this work is split between web infrastructure for mapping and experimentation tools for researchers. The MapTube components are all web-based mapping, with this thesis presenting the theory behind the performance enhancements that make this work in practice. The end result is a system for casual users to find and explore geospatial data, and, hopefully, upload some of their own. The data store mining and MapTube Explorer projects are aimed at academic researchers as they address

the problem of how to get more out of web-based maps than simply visualisations of data. These projects come from the idea of handling sets of geospatial data together as a block, where comparisons can be performed between maps to find similarities. Neither project is designed as a fully working package, as, while they do work on the examples shown here, their worth is in showing the methodology which other researchers can pick up and modify to mine different data stores and perform other types of map comparisons. The value in the code is the generalisation of the problem, so only a small percentage of code is needed to adapt them to a different problem.

Real-time data also falls into the infrastructure category due to the nature of the systems involved. Different logins are needed to access the services for different sources of data. The result presented in this thesis is a system similar to TransportAPI [1], two years before TransportAPI launched. In fact, the ANTS project and subsequent developments in this thesis put forward a more integrated approach to city scale data than just the transport system. This is aimed at researchers, providing simple access to data via an API which also feeds into other real-time visualisations of city data, for example, *http:\\citydashboard.org*, the iPad wall installed in the London Mayor's office in 2012 and an exhibit at the London Transport Museum, along with various other one-off exhibitions and conferences. The main aim with the real-time part of this thesis is to make it easier for researchers to access this type of data needed for research, by collecting everything in one place.

[1] TransportAPI is a public API providing real-time transport data in the U.K. see: *https:\\transportapi.com*

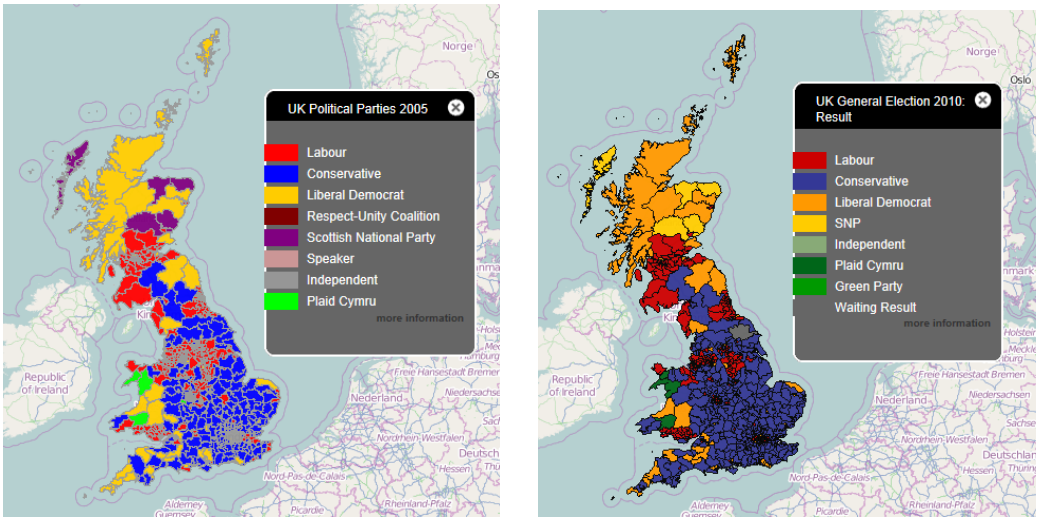
Chapter 2

Literature Review

The following chapter presents a brief overview of the literature relevant to this thesis.

2.1 Presentation of Geospatial Data

The first use of the term, “Choropleth” is by Wright in his 1938 article on population mapping [Wri38], although Tobler points out that the terms “Choropleth” and “Cartogram” were used interchangeably around that time [Tob04] and that the first use of choropleth maps is thought to be by Minard, although Wright was the first to use the name. The choropleth map uses coloured areas to represent data and is one of the most common forms of geographic data visualisation in use today. A good example are the political maps of election results, where coloured areas represent the easily identifiable colours of the political parties who won the majority of votes.



(a) UK General Election Result 2005

(b) UK General Election Result 2010

Figure 2.1: Two choropleth maps of the UK general election. The 2015 and 2017 results could be added to this sequence.

The two choropleth maps in figures 2.1a and 2.1b show the UK General Election

result for 2005 and 2010. Political parties in the UK have standardised colours, so the first problem with the choice of a representative set of map colours is eliminated. People viewing these maps should already be conditioned to accept that red is Labour and blue is Conservative, but this also highlights the main criticism of this type of choropleth. In 2005, red won a clear 50% majority, while in 2010, blue and yellow had to join together to form the required 50%. This is not immediately obvious from the map as the Parliamentary Constituency areas differ in size, giving more visual prominence to those with greater area. Techniques like hexagon maps [Hen10] can avoid this problem, but it is highly dependent on the data being displayed. In the case where the map is used to display outliers on a map that is mainly white, this problem can be turned into an advantage. Also, when displaying Census data, where areas have already been normalised to all contain approximately the same number of people, this might not be an issue.

Openshaw first coined the term, "Modifiable Areal Unit Problem", or "MAUP" in his article, "The Modifiable Areal Unit Problem" [Ope84], even though the problem of zone aggregation in the correlations of U.S. Census statistics had been published as early as 1934 by Gehlke and Biehl [GB34]. Statistical effects caused by the choice of which electoral wards form the different Parliamentary Constituencies is used as an example by Openshaw, who comments on how the zonal aggregation choice in Camden can change the result. In his example, there are 520 Parliamentary Constituencies in 1983, but this has now increased to 650 in 2019, which is reflected in the two maps in figures 2.1a and 2.1b. Openshaw's statement on page 7 of [Ope84] where he says, "the availability of fast super-computers opens up the possibility of seeking approximate numerical solutions" and his comments on "Monte Carlo optimisation methods" in the conclusion both relate to his analysis of MAUP as a combinatorial problem. Hennessy and Patterson state that, "...the highest-performance microprocessors of today outperform the supercomputer of less than 10 years ago" [HP11, pp2], which needs to be taken in the context of the PDP370 that Openshaw was using in 1983 for his analysis. Current desktop computers are 3.5 decades removed from this, or more than 3 supercomputer generations ahead. The graph of processor speeds that the authors use to justify this claim points to a 10,000 times increase in speed between 1983 and 2006. This is significant in the context of this thesis as the aim is to provide a next generation of tools for geographers and spatial analysts to use. With the MAUP problem,

this amounts to a sensitivity analysis to guard against any zoning bias in the results. Either on the desktop, or in the cloud, the compute power now exists to perform this type of analysis. However, the number of zones of interest has also gone up, often with models of the whole of the U.K. containing many thousands of zones in an attempt to combat edge effects. This is an $\mathcal{O}(n^2)$ problem due to the combinatorial nature of the zones, which suggests that compute speed is not keeping up with the requirements if it only increases linearly over time. A robust sensitivity analysis with large n is likely to require a degree of parallelism to compute in a reasonable time.

Presentation of map based data to humans is fraught with numerous difficulties linked to the perception of coloured data. In "ColorBrewer in Print: A Catalog of Color Schemes for Maps" [BHH03], Brewer studies how humans interpret data presented on maps by asking volunteers to answer map-based questions with a variety of colour schemes and under different lighting conditions. This paper forms the basis of the "ColorBrewer" set of colours which is built into numerous GIS systems for example "GeoTools", "QGIS" and "ArcGIS".

Before drawing a choropleth map, though, the colour scale needs to be defined, in other words, how the data maps to a discrete set of colours.[1] In the work by Brewer, [BHH03], she categorises the data according to three types: 'sequential', 'diverging' and 'qualitative'. Sequential data is characterised by continuous real number values, for example a carbonmonoxide sensor value that varies from 0 parts per million upwards. Diverging data is where there is a natural break point, for example, male:female population ratio where the break is at zero. Qualitative data is where the data fits a set of distinct classes where there is no ordering relationship, for example types of dwelling classified as detached, semi-detached or terraced.

In addition to the type of data being represented, the number of colours to use in the colour scale and the data breaks also need to be chosen. These are related in that there must be enough colours for the number of breaks chosen, but how the data breaks are distributed over the range of the data is also required. Firstly, though, the question of how many breaks to use must be answered. In "Comparing continuity and compactness of choropleth map classes" [Cal18], Calka analyses six different methods of classification breaks, identifying the strengths and weaknesses of each. In particular, the inclusion of "Head-tail Breaks" for data that is "not normally distributed" is potentially in-

[1] The colours in the scale could also be defined continuously, for example a linear transition between blue and red based on interpolation of RGB values, but the point being made here is that the blue and red still need to be chosen.

teresting when working with scale-free network data. The remainder of the breaks classifications are: Equal Intervals, Quantile, Standard Deviation, Natural Breaks (Jenks) and Geometric Intervals. In addition to this, there is a review of "goodness of fit" measures, including Jenks' own assessment criteria for his natural breaks classification first published in "Optimal Data Classification for Choropleth Maps" [Jen77]. The paper concludes with examples of population density in Poland, measuring the "Tabular Accuracy Index" (TAI) with reference to two new index methods, "Spatial Distance Index" (SDI) and "Spatial Contiguity Index" (SCI). The conclusion made here is that, "depending on the selection of class ranges in mapping population density, rural areas can be made prominent or the focus can be directed at urban areas with small, medium and large town and cities".

In light of these issues with the visual perception of maps, none of the data store mining (section 4.4.2) or map comparison (section 5.2) actually visualises any maps, though. The only visualisation is as a first guess for humans, which can then be altered. All the map comparisons and data store mining presented in this thesis is visualisation agnostic. All the data is machine to machine, up to the point where a human sees a meta-visualisation of the map comparisons e.g. a graph of linkage.

2.2 Agent-based Modelling for Real-time Data

Chapter 6, "Real-time Mapping and Agent Simulations" uses the technique of agent-based modelling to construct models of real-time transport networks. Crooks and Heppenstall provide an introduction to agent-based modelling in "*Agent-Based Models of Geographical Systems*" [Hep+12, Ch5], including geographical systems. Epstein and Parker used agents in a simulation of the global dynamics of disease propagation, which is notable for its use of one agent for every person alive in the real world [PE12] and [EP09]. While this example simulates millions of agents using a custom application, "FlameGPU" [She19] is a GPU implementation of a large-scale agent based modelling library. Similarly, the commercial company "Improbable" also market their "SpatialOS" [Imp18], which is targeted at large-scale agent-based modelling for the computer games industry, being based around the "Unity framework" and load-balancing agents on grids using cloud computing. The issue of asynchronous simulations, ordering of operations and conflict resolution needs to be handled. Richmond uses a system of serial transaction processes to mitigate conflicts in "Resolving Conflicts between Mul-

tiple Competing Agents in Parallel Simulations" [Ric14], thereby solving the problem for his computation of agents on a regular grid structure. Here, the problem is where agents move from one grid to the next as parallel processes could result in two agents moving to the same grid cell, which is not allowed. The solution involves detecting where conflicts occur and using a priority ordering scheme to repeat the decision process for a lower priority agent that attempts to move onto a grid cell about to be occupied by a higher priority agent. The author makes the point that the performance of the implementation is dependent on the application[2] and the number of conflicts that occur in what is a transactional based solution.

A more general introduction to the process of modelling can be found in Wilensky and Rand's "Making Models Match: Replicating an Agent-Based Model" [WR07]. Here, the authors put forward the case for "replication" in models, citing agent-based models used in scientific papers that have never been replicated outside of the original publication. The document can be seen as a set of "best practices" for model authors, designed to ensure the replicability of the models by other researchers using different computer languages, systems and practices. The key point they make is that the rules behind the model should be explicit and open to scrutiny.

The research presented in Chapter 6 outlines an agent-based model of the London Underground and the London bus network. This was influenced by Rand's work, "Machine learning meets agent-based modeling: When not to go to a bar" [Ran06]. He suggests the idea of learning the rules of the model by observing a running system, an idea which is adapted to "learn" the operation of the London Underground by observing the data available on the real-time stream.

2.3 Analysing Tube and Bus Strikes

In chapter 8 the analysis of bus and tube strikes is used as a worked example to show the general usefulness of the real-time software developed to that point. The London Underground is used as an example by D'Lima and Medda in "A new measure of resilience: An application to the London Underground" [DM15], where the resilience of the system to shocks is investigated. They adopt the measure of, "the speed at which a system returns to equilibrium after a disturbance away from equilibrium", using data from the real-time stream on tube positions, along with passenger flow data.

[2]The application used for the demonstration is Epstein and Axtell's "Sugarscape" example in their book, *Growing Artificial Societies: Social Science From the Bottom Up* [EA96].

The methodology works on the basis that a delay on one of the lines will lead to a sharp decrease in passenger flow. In addition, the authors take into account minor delays on connected parts of the network as a propagation of the initial shock. They take the resiliency of the line to be a factor in a model based on Brownian motion which is fitted to the passenger flow data. In their analysis they state, "the available data was adjusted at the source to remove the effect of abnormal circumstances that may affect passenger demand such as industrial action". They conclude with an analysis of the system under different types of shock.

On the direct results of a tube strike on the London Underground, Larcom, Rauch and Willems have published, "The Benefits of Forced Experimentation: Striking Evidence from the London Underground Network" [LRW15], stating that "a significant fraction of commuters on the London Underground do not travel their optimal route". The paper then goes on to demonstrate evidence to suggest that a result of the strikes in 2014 was to force many commuters to try different routes, whereby they discovered a more efficient one. Their paper is based around the decisions commuters make in the face of imperfect information, using Oystercard travel data from Transport for London as their empirical evidence. This data is multi-modal, including tube, train, tram, Docklands Light Railway (DLR) and riverboat services, but their study only focuses on the underground data and repeat journeys made by commuters after the 48 hour strike has finished. Analysis is by fitting the data to a regression model that they propose and deriving the coefficients from the data. Toward the end of the paper, the question of travel time is more relevant to this thesis, but the authors state an obvious problem with the use of Oystercard data. Not all commuters "tap in " and "tap out", when they enter and exit the system[3] and when changing from one tube line to another. Even when an entry/exit record can be matched for a commuter, the path they took through the network can only be estimated using a shortest path, or most probable path, algorithm. The conclusions of the paper are that, "the tube network was operating so far away from its optimum, that February 2014 strike managed to improve efficiency of the system as a whole". This is quite a bold claim to make, especially in light of the fact that the paper does not consider the effects of loading and capacity in any way. The authors finish by suggesting that strikes are quite extreme ways to encourage commuter change, but improved forms of information dissemination (e.g. "journey planner apps") can "nudge

[3] Season ticket holders can simply walk through the barriers if they are open and no entry/exit data for them is collected. During periods of heavy demand, barriers are often opened for public safety reasons.

travelers to experiment more".

2.4 Combining Real-time and Static Data

The combination of real-time data with static data can take a number of different forms. Firstly, the concept of "polystores" is gaining momentum in the data management and semantic knowledge representation journals, for example the "ACM Special Interest Group on Management of Data (SIGMOD)"[4].

While combining real-time and static data encompasses polystores and data warehousing technology, in "High resolution population estimates from telecommunications data" [Dou+15], Douglass uses real-time telecommunications data to derive population estimates which are then verified using Census data. They also find that they can make estimates by age, gender and ethnicity, but only by using a model which is calibrated on the specific area of study. They warn against trying to apply the same results to another area without any base data to build the model from. Their conclusion states that, "We envision an inter-census calibration using a very small scale stratified population count in key calibration regions", making their point that the model is a way of augmenting the existing static population data. While this example uses data in an offline sense, real-time streaming can often require the processing of data sets that grow rapidly over time. Architectures like the Lambda Architecture and Kappa Architecture, along with high performance computing like Apache Spark are often utilised for this task due to the volume of real-time data. Central to the concept of the Lambda architecture[5] is the base transaction data, which is analysed using a batch, speed and serving layer. Nathan Marz is credited with inventing the term, defined in his book, "*Big Data: Principles and best practices of scalable realtime data systems*" [MW15].The general idea behind the architecture is that data is processed quickly and approximately to provide immediate results on one thread, while on a parallel thread, the exact result is being computed using a different method taking a longer time. The serving layer allows queries to be made on the basis of the best data available at the time.

As the size of datasets increases, so algorithms that act on data as a stream operation are becoming more popular. The basic idea follows the data stream learning principles described in Chapter 9 of [WEH11]. The following quote defines the methodology:

> "One way of addressing massive datasets is to develop learning algo-

[4] SIGMOD: *https://dl.acm.org.sigmod*.
[5] The Lambda Architecture is defined at: *http://lambda-architecture.net*.

> rithms that treat data as a continuous stream. In the new paradigm of data stream mining, which has developed in the last decade, algorithms are developed that cope naturally with datasets that are many times the size of main memory - perhaps even infinitely large. The core assumption is that each instance can be inspected once only (or at most once) and then must be discarded to make room for subsequent instances."
>
> (from *"Data Mining"* [WEH11, pp380])

In other words, the memory footprint of the program remains constant as the data is fed through it. The power of this method is in its applicability to very large datasets where an answer that is approximately correct using frugal compute resources is preferred over spending more time and effort on an answer which is only a few percent more accurate.

The "Frugal Majority" algorithm [Boy91], shows this frugal paradigm in action when calculating whether the stream contains a class with more than a 50% majority. The more obvious way of writing this would be simply to count how many of each class there were, but this would have a memory footprint that increased with the number of classes in the data. While the majority class algorithm is a simple example, the "Count-Min" algorithm [CM04] is a probabilistic method similar to a Bloom filter [Blo70], but which applies multiple hash functions to data in order to count the different values. The importance of this algorithm is that the technique can be used to find approximate quartile ranges on very large datasets using a single pass through the data and no sorting. The technique relies on having enough hash functions so that a hash collision between two data values in one hash table is unlikely to result in a collision on all hash functions simultaneously. The histogram of data values encountered is then calculated from the data in the hash tables.

Finally, the combination of static and real-time data is being investigated by the Office for National Statistics (ONS) Big Data team researching the use of mobile phone data[6] for population estimates, urban planning, commuter flows, ethnicity and community, amongst other areas. While much of the work centres on mobile phone data, for example [Dou+15], earlier, and "Redrawing the Map of Great Britain from a Network of Human Interactions" [Rat+10] looking at regional boundaries, and "Poverty

[6]The ONS report is available at: *https://www.ons.gov.uk/methodology/methodologicalpublications/generalmethodology/onsworkingpaperseries/onsmethodologyworkingpaperseriesno8statisticalusesformobilephonedataliteraturereview*.

on the Cheap: Estimating Poverty Maps Using Aggregated Communication Networks" [SMC14], looking at poverty maps. The key driver here is the availability of a real-time feed and a static data set to use for the baseline calibration.

2.5 Computer Graphics and Rendering

The myriad different ways that data can be visualised on a map makes attempting to compile an exhaustive list impossible. The current popularity of "data graphics" or "infographics" highlights this seemingly insatiable desire for innovative visualisation methods. For this reason, the emphasis must be placed on systems which facilitate the creation of visualisations generally, rather than looking to support a subset of the most popular ones. Edward Tufte's book, "The Visual Display of Quantitative Information" [Tuf83] defined a 'Lie Factor' metric and introduced the concept of 'Chart Junk'. His rules for the principles of graphical excellence and graphical integrity of visualisations apply to maps as well as general data visualisations. Rather than quote all his rules verbatim, one particular quote sums up data visualisation quite succinctly:

> "Above all else, show the data." (E. Tufte 1983)

In terms of contemporary data visualisation, many websites and blogs have appeared which allow the uploading, creation and sharing of data visualisations. IBM's 'Many Eyes' website [IBM07], which was launched as an experiment in January 2007 and ran until 12 June 2015, before being closed in favour of 'Watson Analytics' is one example. While not strictly a spatial visualisation site, there is a cross-over between general data visualisation and spatial data visualisation. Following on from IBM's experiment, other websites have appeared, for example, Nathan Yau's 'Flowing Data' blog[7] and two books, [Yau11] and [Yau13], on data visualisation, which are direct spin-offs from his PhD thesis on personal data collection from the University of California. Mike Bostock's 'Data Driven Documents', or 'D3', library [Bos16] facilitates data visualisation on web pages using Javascript, HTML5, Canvas, Scalable Vector Graphics (SVG) and other Document Object Model (DOM) elements. This is a development of the original 'Protovis' library, but using a functional programming style, where data is transformed into a visualisation on the page, much like XSLT transforms XML data into XHTML web pages. While the functional style does work for some types of data, it can be cumbersome when transforming spatial data using map projections. Styles are used

[7] Flowing Data: *http://flowingdata.com*.

to control the appearance of the data on the map, in a similar way to how 'map CSS' is used to style cartography on tiled maps (*wiki.openstreetmap.org/wiki/MapCSS/0.2*). Using the 'D3' library, data on maps can also be animated, or data from sensors can be displayed in real-time.

Geographic visualisation generally, can be seen as methods for displaying point, line and polygon data. For example, Shepard interpolation using inverse distance weighting [She68], contours [Far86], choropleth maps [Wri38], cartograms [Dor96], heat maps, flow maps [Pha+05] and spatial correlation methods [Ans95]. The detail of how to place the data onto maps is covered in Chapter 4.

2.6 Automatic Data Store Mining

Google's Fusion Tables web service is one product for visualising data on the web, having been released in 2009 as an experimental service. In the only publication written on how the system works, "Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud", [Gon+10], the authors show how the 'BigTable' database stores the data in an unstructured format using key value pairs using indexing with a row store and schema store for all the tables. In the context of automatic data store mining and geographic data, Fusion Tables has the ability to determine the geographic context of rows of data and insert "geo" objects (points, lines, polygons) based on geocoding the data. While this allows users to make maps from data, it is not an automatic mapping system for data generally. This requires a tool that can go through all of the maps stored on Fusion Tables, extract the spatial context of the data, and perform a geospatial operation like clustering together all the maps that are similar. Other systems like 'BatchGeo' (*http://batchgeo.com*) and 'Tableau' (*http://tableau.com*) exist which do a similar job of making maps from data, but only on a single data set, while MapBox (*http://mapbox.com*) and ArcGIS Online (*http://esri.com*) both perform similar functions for web based maps, with ESRI also allowing integration with its ArcGIS desktop platform.

The work presented in this thesis provides similar functionality to the work just reviewed, in that it allows users to make maps from data more easily than before. Blocks of data in CSV files can be uploaded and identified using similar techniques to those presented in [Gon+10], using various data mining methods to extract the spatial context of the data, find a column of data values to map and pick suitable data breaks. However,

there is a difference between Fusion Tables and how the MapTube [8] system presented here works in that MapTube leaves the data on the web, while Fusion Tables is also a data warehouse. This thesis presents work that exploits this difference and shows how to use an automatic mapping system to handle data from multiple sources and perform geospatial operations like comparing and clustering maps. This is a new topic which does not appear in any literature, so must be broken down into its component parts for review.

Firstly, identifying the type of the columns in the data is a "Geospatial Data Mining and Knowledge Discovery" task. Gonzalez describes how they achieved this as follows: "Rather than having the user declare a schema for the tabular data and then having the user also describe how the data to be imported matches the schema, the system tries to detect automatically which row in the imported file is the header row" [Gon+10]. No technical information about how this works exists. Similarly, in "Many Eyes: a Site for Visualization at Internet Scale" [Vie+07], which presents the early development of the "Many Eyes" website for sharing data visualisations, the same problem of identifying the type of data columns exists. The authors state:

> "A number of commercial systems have begun to explore the idea of asynchronous communication around data. Several web sites (e.g., Dataplace, Data360, Swivel, DabbleDBand Chartall) allow users to upload and graph data. Swivel, Data360, and Chartall allow users to make comments; Swivel especially seems to aim at scaling to a large audience, styling itself as "YouTube for data". "
>
> ("Many Eyes: a Site for Visualization at Internet Scale" [Vie+07])

Actual detail on how any of these systems work is impossible to ascertain without looking into the development in detail (if possible). The reason this quote has been included is to make the point that they are all leveraging existing software libraries without necessarily developing something new or innovative. Their net worth is in the collection, not the details.

In "Table structure understanding and its performance evaluation" [WPH04], Wang applies a probability based approach to the problem of table structure understanding. While this branch of pattern recognition techniques seek to give structure to documents as a whole, by identifying tables of data and blocks of text within them, this goes further

[8] MapTube was launched in February 2008 and pre-dates Fusion Tables, which was launched in June 2009.

than required for data uploaded to a server as is the case with Fusion Tables and Many Eyes. Only the column identification part is required, based on the statistics generated from analysing the table. Wang states that, "the algorithms can be classified into three categories: (1) pre-defined table layout based; (2) heuristics based; and (3) statistical or optimization based", before going on to review the techniques as used in a number of other papers.

On the problem of automatically finding data breaks, this was covered in the previous section 2.1 on presentation of geospatial data and is only relevant if the data is being built into a map visualisation. Only the area key and data value columns are needed to perform most spatial analysis algorithms, for example, cross correlation of two maps, or all permutations of maps in a data store to find similarities. Once the automatic mapping has got to this stage, then it becomes a geospatial data mining problem.

2.7 Linking GIS and Computer Graphics

Exploratory spatial data analysis (ESDA), as distinct from GIS, is defined in "Interactive techniques and exploratory spatial data analysis" [Ans99]. Here, Anselin is describing interactivity going beyond a regular GIS system, where users can explore and question data, relying on computer graphics for interactive visualisations. He makes the distinction between static techniques, like spatial autocorrelation, and dynamic, interactive, techniques, where the user can interact with the data directly in a graphical environment. He goes on to say,

> "The main contribution of ESDA with respect to GIS lies therefore in visualising local patterns of spatial association, indicating local non-stationarity and discovering islands of spatial heterogeneity."

This relates ESDA to GIS, but Chrisman, in his book, "*Charting the Unknown. How Computer Mapping at Harvard Became GIS*" [Chr04], chronicles the development of GIS from computer graphics. Containing details about the "SYMAP" program from 1963, in the "how SYMAP worked" section, Chrisman says, "SYMAP used a vector model, a collection of objects-points, lines, and areas-in planar coordinate space with thematic values attached to them". The same is true today, although with the possible exception of bezier patches, point clouds and voxels (all defined in [Fol+93]). Maguire puts forward a definition for GIS in [Mag91], proposing three distinct views which define a GIS system: "map, database and spatial analysis". In his definition, "each

data set is represented as a map", which we extend in this thesis to become a data set representing a collection of maps. His definition of "database view" refers to storage, which could be cloud-based, while the "spatial analysis view" is the visualisation layer. He goes on to describe variants of GIS software called, "file processing, hybrid and extended designs". The file processing design pattern stores data sets as separate files, which are linked together later in the processing pipeline. The hybrid approach is similar, but splits attribute data (aspatial) and spatial data. The extended model is similar to the hybrid, but extends the relational database layer with spatial extensions. Of these three approaches, the idea of holding data in the cloud and joining with geospatial data in a different cloud location, while running spatial and aspatial data through different systems would seem to lie somewhere between the file processing and hybrid approaches. This is the approach taken by the MapTube system.

As far as computer graphics goes, this needs to be taken in a web based context, where the computer rendering the data is distinct from the computer that processed the data. In this model, the computer browser does the rendering, which makes for a distributed GIS. Shaowen Wang advocates this type of GIS system, under the definition, "cyber-infrastructure" [Wan10], and in the "PYSAL CyberGIS Toolkit" [Rey+13]. Wang describes this as "software as a service", offering GIS functionality as a web service.

Two approaches to rendering maps in a web browser exist: raster or vector. Much depends on the capabilities of the computer browser, but raster involves the server rendering the map tiles, while the vector technique sends scaled, clipped and tiled versions of the data to the browser which then has to render the data using WebGL [Khr14]. Either way, the rendering involves computer graphics algorithms to draw the maps from points, lines or polygons, following the OGC's SFS standard [Ope10b].

Advances in modern graphics hardware now mean that computers have a graphics processing unit (GPU) in addition to a CPU, which provides specialised hardware designed for rendering. To this end, the vertex shader and pixel shader hardware [CR11, Ch3] are used to accelerate the computer graphics. In "A Lightweight CUDA-Based Parallel Map Reprojection Method for Raster Dataset of Continental to Global Extent" [LFB17], Li shows an example of reprojection using the CUDA library designed by NVIDIA to run on their GPU hardware. The authors also point out the CyberGIS link in their introduction:

> "To solve these types of computationally intensive problems, previous approaches usually relied on the use of high-end workstations, thus making efficient map reprojections for raster datasets less accessible to scientists who do not possess expensive infrastructural resources such as high-end computer clusters or supercomputers."

On the client side, the MapBox library (*https://www.mapbox.com/mapbox-gl-js/api/*) uses WebGL to render maps, while 3D applications in the browser have also been created, for example "OpenWebGlobe" of West et al. [Wes+14]. Rendering using pixel and vertex shaders can have some interesting applications, for example photo-realistic renderings of the Earth, ray tracing and procedural modelling. In "Real-Time Rendering Techniques with Hardware Tessellation" [Nie+16], the authors summarise work on rendering techniques for hardware tessellation, including parametric surfaces and real-time tessellation of surfaces. In the paper they analyse height fields and terrain rendering, which are relevant for 3D work, including the minimisation of splits and cracks caused by the tessellation, a subject which Cozzi and Ring also cover at length in their book on the development of an OpenGlobe system [CR11].

2.8 Comparing Map Data and Correlations

The visual similarity of maps and the effect that mapping has on the human perception of the data has been explored in various literature, for example, in "Visual and Statistical Comparison of Choropleth Maps" [LS77], Lloyd and Steinke look at the Pearson correlation coefficients of the raw data, framing the process of mapping in the context of adding noise to the data. They conclude with the comment that,

> "If the concept of optimal maps is ever to become a reality, a complete under-standing of all map attributes and their effect on the visual similarity of maps will be a necessary pre-requisite."

In their user experiment, they use "equal interval", "minimum deviation" and "equal area" variations of the same data for comparison. The error introduced by these data breaks is treated as if it was noise and the raw data as signal, although this form of visualisation can also be seen as a reduction in information. In "Associations in Choropleth Map Comparison" [Mul75], Muller also looks at map comparison, stating that, "The main objective of map comparison is to discover correlations of factors geographically.". Where Lloyd investigated human perception and the visual effects of mapping,

Muller takes a statistical approach to comparison. Although fundamentally different papers, Lloyd's analysis of noise hints at a more mechanistic approach to the comparison of maps, following Muller's method of looking for similarities in the raw data. The idea that the choropleth map is a visual presentation for a human audience, while the associations between maps can be expressed mathematically is expressed by Muller as,

> "The relevance of choropleth map comparisons for discovering correlations or more generally mathematical relations between geographic variables depends main on the readers' ability to visually associate maps whose "look" is different but whose pattern is identical." (Muller [Mul75])

His paper uses various test maps of France for positive, negative and cross relationships under linear and non-linear conditions. The analysis depends on the map being seen as a mathematical function on the data, $f(Z)$, with analysis of the correlation providing information on the functional relationship between two maps. The mathematical treatment of the comparison divides into correlation, which is the relationship between the underlying data sets, and spatial distribution, which is where the data is positioned on the map. This is an important distinction to make, which is demonstrated in the paper through the displacement of a pattern of points over France. In order to look for linear correspondence, the Z variables for two maps are ranked and compared. A linear relationship is defined where there exists an α such that $Z_1 = \alpha Z_2$, which can be either positive or negative. However, the ranking process has removed the spatial context, which Muller handles separately via a network graph clustering method. Contrast this to "Multivariate Spatial Correlation: A Method for Exploratory Geographical Analysis" [War85], where Wartenberg defines a spatial cross correlation coefficient based on Moran's I:

Wartenberg Bivariate Cross Moran

$$I_{xy} = \frac{n}{\sum_i \sum_j v_{ij}} \frac{\sum_i \sum_j v_{ij}(x_i - \overline{x})(y_j - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2} \sqrt{\sum_j (y_j - \overline{y})^2}} \tag{2.1}$$

where

I_{xy} Moran I cross correlation coefficient, $I_{xy} = [-1..1]$

i, j area indexes into the x and y datasets, where $i = j$ references the same area in each

x_i data for area i in dataset x

y_j data for area j in dataset y

$\overline{x}$ mean of x values over all i areas

$\overline{y}$ mean of y values over all j areas

v_{ij} weighting factor between area i and area j, typically [0..1]

Here, data is normalised by mean and standard deviation, with spatially derived weights. The spatial and data relationships are both treated together, resulting in a formula that is of order $\mathcal{O}(n^2)$ for n zones. Computationally, this is significant when the number of zones is 7,201 for MSOA level data, while Muller's data relationship tests were $\mathcal{O}(n)$. However, Muller's spatial relationship tests are based on graph comparisons and are the same order computationally as Wartenberg's cross correlation formula. The question remains about whether a fast relationship test of $\mathcal{O}(n)$ is a useful heuristic for finding maps that are *potentially* related, before following up with a full analysis of the type of relationship. Empirical results are likely to be needed to verify if this is the case. Looking at the spatial cross correlation formula, the issue of how to determine the weights needs to be solved. In Anselin's workbook for his Geoda software for spatial analysis[9], a number of possibilities are explored, for example, binary weights to indicate a neighbourhood relationship, or real-valued weights to represent distance. One other problem exists, which involves the spatial autocorrelation of data. In "Spatial Autocorrelation: A Review of Existing and New Measures with Applications" [CO70], Cliff and Ord examine spatial correlation, using the Moran I [Mor50] and Geary C [Gea54] tests for spatial autocorrelation of maps. The issue occurs with the cross correlation of data that is spatially autocorrelated.

Returning to the original issue of map comparisons, testing every combination of map pairs using a mathematical comparison would appear to solve the problem of perception, which Lloyd investigated earlier in [LS77]. Two problems still remain, though, regarding what comparison is best to use and how do the computation in a reasonable amount of time. Taking the method of comparison first, as the method itself is part of

[9] *https://geodatacenter.github.io/workbook/4a_contig_weights/lab4a.html*

the research question, any solution should able to use a variety of methods. The papers written comparing different methods make that clear: [Mul75], Geary C [Gea54], Moran I [Mor50], Spatial Autocorrelation [CO70], Wartenberg [War85] and Anselin's LISA [Ans95]. The second problem is a computational problem involving the optimisation of nested $\mathcal{O}(n^2)$ loops. As stated previously, the spatial autocorrelation function is $\mathcal{O}(n^2)$, while the number of permutations of any pair of maps (m) is:

$$Permutations(m) = \frac{m(m-1)}{2}$$

(Assuming that the correlation function is commutative)

So, the time to run a spatial autocorrelation on all of the maps in a set is $\mathcal{O}(m^2 \times n^2)$. Similarly, to add a new map to an already existing set of correlations is $\mathcal{O}(m \times n^2)$. One possible way forward could be to match against a smaller number of templates as a heuristic. The only problem here is that if Map A correlates highly with Map B, then Map C is not guaranteed to correlate with Map A, even if it is highly correlated with Map B. This could be formulated as a spatial feature extraction algorithm. Gangappa et al. provide a review of regression, Bayesian, Artificial Neural Network and other ensemble techniques in "Techniques for Machine Learning based Spatial Data Analysis: Research Directions" [GMS17]. Openshaw's book, "*Artificial Intelligence in Geography*" [Ope97] provides a good introduction as well, but the technology has moved on since it was written in 1997.

Given the nature of the problem defined so far, the use of parallel algorithms to accelerate the processing is a possibility. Artificial Neural Networks (ANNs) have gained popularity recently with libraries like TensorFlow [Aba+16], Keras [Cho15], PyTorch [Pas+17] and Caffe [Jia+14] providing interfaces to utilise the multiple parallel cores of graphics processing units (GPUs) for large-scale networks. These ANN simulations use matrix operations (hence "TensorFlow"), running on "general purpose GPU" libraries. The libraries are not so much specialised to the task of simulating ANNs as to the task of running matrix operations and so are usable in the context of large scale spatial correlations.

2.9 Learning from Real-time Streams

One definition of learning is "Hebb's postulate of learning", from *The Organization of Behaviour: A Neuropsychological Theory* [Heb49, pp62], in which he defines the

process of learning as follows:

> "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased." (Hebb, 1949 [Heb49])

This defines the term, "Hebbian Learning", or learning by reinforcement. When data from real-time streams is considered, this reinforcement comes through the repeating patterns present in the data. Reinforcement learning is the topic of ""General Principles of Learning-Based Multi-Agent Systems"" [WWT99], where Wolpert et al. describe the optimisation of systems in the absence of any centralised control. To quote their justification for using the agent-based approach as opposed to a "hand-tailored design":

> "one does not have to laboriously model the entire system; global performance is"robust"; one can scale up to very large systems; and one can maximally exploit the power of machine learning."
>
> (Wolpert et al. [WWT99])

The authors state that this approach is also applicable in the fields of: "multi-agent systems, computational economics, reinforcement learning for adaptive control, statistical mechanics, computational ecologies and game theory". This highlights the fact that the techniques of learning by reward, or maximising a utility function, are generally applicable to a wide variety of problems.

From a computational point of view, the following quote by Aleksander and Morton from the book, "*An Introduction to Neural Computing*" [AM95], reinforces the point that Wolpert et al. make, but frames it in a wider perspective:

> "The art of the computer programmer is that of turning algorithms into code that the computer can understand through the use of a suitable computer language, while the art of the AI scientist is to find algorithms for the forms of computer behaviour he or she is trying to achieve."
>
> (Aleksander and Morton [AM95])

This is put into practice by Rand, in "Machine learning meets agent-based modeling: When not to go to a bar" [Ran06], where he develops the idea of learning models from real-time streams. The example presented draws on the same problem cited by

Wolpert et al. [WWT99], which is Brian Arthur's bar attendance model [Art94]. Rand, however, includes the concept of an "agent world" and a "real world", which the model attempts to replicate through modification of its rules. Related to this idea, Kirman and Vriend study the working of the Marseille Fish market in "Evolving market structure: An ACE model of price dispersion and loyalty" [KV01] and "Learning in Agent-based Models" [Kir11], which they describe as an "agent-based computational economics" (ACE) model. The authors stress the use of the word "computational":

> "The important word here is "computational". The idea is to specify the nature of the agents involved, the rules by which they behave and by which they interact and then to simulate the model in order to observe the outcomes." (Kirman [Kir11])

Kirman and Vriend use the data from the market transactions to learn decision probabilities for their logit model. Reference is made to game theory and Holland's complex adaptive systems, which most closely follows how their model is developed, being composed of "IF THEN" rules, similar to Holland's genetic agents [Hol95]. The aim is to find the free parameters of the system to optimise the fit with the real world data.

Relating learning to physical systems, in "Why does deep and cheap learning work so well?" [LT16] Lin and Tegmark make the following claim:

> "We have shown that the success of deep and cheap (low-parameter-count) learning depends not only on mathematics but also on physics, which favors certain classes of exceptionally simple probability distributions that deep learning is uniquely suited to model." (Lin [LT16])

The paper relates to how network size and structure determines the types of problems that can be solved, concluding with the statement above, that the speech, language translation and object recognition problems that currently work so well all belong to a narrow class of physical problems. A review of the field of deep learning and artificial neural networks in relation to agent-based modelling is given by van der Hoog in "Deep Learning in (and of) Agent-Based Models: A Prospectus" [Hoo17], including a history of the field. The authors' approach is a form of competitive learning, which they term the "Doppelganger approach". Agents imitate and try to out-perform their peers, who they then replace in the simulation. However, the "maximisation of utility function" from previous approaches is substituted with an alternative metric based on

performance. The authors' secondary aim is directly related to learning models from streams of data:

> "...we propose to use ANNs as computational emulators of entire ABMs. The ANN functions as a computational approximation of the non-linear, multivariate time series generated by the ABM. It is a meta-modelling approach using statistical machine learning techniques."
>
> (van der Hoog [Hoo17])

The problem of model validation is mentioned, but van der Hoog also states that, "...until now, no clear consensus has appeared how to resolve the the empirical validation problem.". The two methods he refers to are bootstrapping [BHM07] [ET94] and "estimation of a master equation derived from Focker-Planck equation" from state transition probabilities [ALW05].

2.10 Knowledge Discovery

Knowledge discovery has already been touched on in section 2.6 in the context of automatic data store mining as there is overlap between knowledge discovery, data mining and machine learning. The Springer journal "Data Mining and Knowledge Discovery" contained a special issue on "Smart Cities", acknowledging the increasing use of city data. In "Structural robustness and service reachability in urban settings" [AZB18], Abbar et al. look at the concept of *urban resilience* in the context of the Rockefeller Foundation's "100 Resilient Cities Program" (100RC). From the data science point of view, they required road network data and the geographic distribution of services from every city in their study. Their analysis proceeds by building networks from the road data and calculating statistics for: vertices, edges, total length, average degree, together with "meshness" and "organic score" which are defined by Wang [Wan15]. The discussion of the results centres on an analysis of the road network graphs. He concludes with the remark,

> "...a systematic study of these "robustness fingerprints" may lead to a classification of cities with regard to their fragility and service distribution imbalances-along the lines of Louf and Barthelemy [LB14], but taking an urban resilience angle."

The article by Louf and Barthelemy, "A typology of street patterns" [LB14], which

is mentioned in the above quote, is another example of network graph analysis applied to the road networks of cities. The aim of both pieces of research is to discover regular patterns in the data, which links to the previous section 2.8 on comparing maps and correlations. Where map data was concerned, Muller also used a graph technique to look for his "associations in choropleth maps" [Mul75], but at a higher level than the road networks being analysed here. Both Wang and Louf make the point that their analysis is only made possible by new data that has become available, which make road network data accessible to them.

Cimini defines a network as:

> "...a network represents the simplest yet extremely effective way to model a large class of technological, social, economic and biological systems, as a set of entities (nodes) and of interactions (links) among them."
>
> (Cimini et al. [Cim+19])

The knowledge derived from analysing networks of this kind is in discovering patterns in the data, but the graph techniques presented here suffer from the problem of computational complexity. The networks can get *big* quickly, for example Wang's analysis of the city of Los Angeles had 525,246 vertices and 678,652 edges. While in a street network this is not too big a problem as the average degree is ≈ 2, in fully connected networks the average degree can grow with $vertices^2$. This is exactly the type of problem being described in section 2.8, "Comparing Map Data and Correlations" and section 2.6 "Automatic Data Store Mining", for comparing all combinations of the maps.

In "The Statistical Physics of Real-World Networks" [Cim+19], Cimini et al. make the point that:

> "Notably, most of the networks observed in the real world fall within the domain of complex systems, as they exhibit strong and complicated interaction patterns, and feature collective emergent phenomena that do not follow trivially from the behaviours of the individual entities."

The authors then go on to introduce *scale-free* networks [BA99], where the node degree follows a fat-tailed distribution or power law and *hubs*, and where a few nodes exist with high degree. A detailed mathematical description of operations on network graphs is given by Dorogovtsev et al. in "Critical phenomena in complex networks" [DGM08]

and also Barabasi in his network science book [Bar16]. Small world networks [SK78], which inspired Milgram's famous "six degrees of separation" [TM69] and Erdos-Renyi random networks [ER59] are also essential to the analysis of the structure of networks.

Finally, on networks, Caldarelli and Catanzaro remark that, "some collective behaviour... cannot be predicted by looking at the single elements forming the system" [CC18, pp2]. Although referring to complex systems, in section 2.6, the data being mined is linked through relationships to place and so needs to be analysed in this context. However, given that a network formed of relationships between spatial data sets has not been constructed from a set of production rules like the other examples in this section, there is the possibility that a network-based analysis of this data could discover something new.

In addition to data mining spatial data and calculating statistical factors that can be presented graphically on a map, there is also the question of how to handle knowledge in the form of the semantic descriptions of what the maps contain. Up to this point only the data on the map has been considered and not the explanation of what the data represents. Without this the data is essentially meaningless, in fact one user of the MapTube site used the phrase, "colouriser", to refer to a set of "pretty coloured maps without any descriptions". This raises a point about volunteered geographic information [GMH15], where users can upload their own data, but might describe the data in a less than rigorous way[10]. One avenue of research is to ask the question of whether it is possible to identify an unknown data set using the context of where it sits in the current framework of knowledge ("blind identification").

Two approaches can be taken to the descriptions of spatial data: free text natural language or a structured semantic description using a well-formed ontology. The idea of using an ontology where users fill out a form with information that describes their data was rejected early on in the development of the MapTube website due to being unworkable. As an example, take a field called "year". Obvious examples are '2019' or '1966', but 'Pre-Cambrian' is a bit more difficult when describing archaeological or geological maps.

The "GeoSPARQL" standard [Ope11], is a geographic query language for "RDF" data. "RDF" is an acronym for "Resource Description Framework" which is a W3C standard for linked data stored as triples composed of: subject, predicate, object. The

[10]While MapTube forces the user to enter a short and long description of every map uploaded, without advanced natural language processing, there is nothing stopping them from entering something like "TBA" (to be added later).

storage of data in this format allows complex queries about the relationships between data to be performed. The "NeoGeo" vocabulary (*http://geovocab.org*) exists as a draft specification, which is an attempt to bring together the currently disparate ontologies currently published by a number of different organisations who curate geospatial data. The Ordnance Survey is one of these organisations, publishing their "Spatial Relations Ontology"[11] which was designed by John Goodwin for querying relationships between geographic areas like "neighbour" and "within". Ontologies also exist for postcodes, administrative areas and the 50K Map Gazetteer (e.g. City, Farm, Forest, Water etc.). In the U.S. the National Science Foundation (NSF) also funded a geospatial interoperability project with similar aims [Wie11].

For an ontology related to the data description, Hochmair examines efficient searching on Internet portals [Hoc05]. Here, the author singles out two standards covering the storage of geospatial data as being important:

- The Content Standard for Digital Geospatial Metadata (CSDGM) version 2 (FGDC-STD-001-1998)
- ISO/TC 211 19115-2003

Hochmair's application uses the "Ontology Web Language" (OWL), which is a W3C standard [W3C05] for ontologies, allowing web crawlers to semantically index data in data stores. The body of the paper is an extension to previous work by Holscher in, "Web Search Behaviour of Internet Experts and Newbies" [HS00], and Pirolli in, "A user-tracing architecture for modeling interaction with the world wide web" [Pir+02], on more general searching of the world wide web. Notably, Hochmair uses natural language English dictionaries of nouns, verbs, adjectives and adverbs from WordNet [Mil95] to build the system that he refers to as "intelligent query expansion". Similar corpus are used in modern sequence to sequence machine learning systems, although on a larger scale, for example in the natural language processing system built in "Smart IoT and Soft AI" [Mil+18] [Mil+] around the Google Dialogflow application[12].

The use of graph tools to build graphs from metadata repositories is the subject of [Ulr+18], where Ulrich et al. analyse health record data using a "Neo4J" database[13]. To quote their justification:

[11] The Ordnance Survey list of ontologies is published at: *http://data.ordnancesurvey.co.uk/ontology*.
[12] Google Dialogflow: *https://dialogflow.com*.
[13] Neo4j: *http://neo4j.com*.

"Summarised metadata itself is a highly connected object which gains its value from meaningful semantic connection to other data objects [EJX01]."

Although the quote is originally from another paper on making connections by attribute matching, the results show the authors extracting structure and new knowledge about the data from the metadata records. What is missing here, however, is a methodology for linking data where the attributes are natural language descriptions. Given the progress made in artificial intelligence for natural language processing in recent years, for example the sequence to sequence deep neural network models of Sutskever [SVL14], there is an increased ability for handling human-readable content without resorting to manual semantic labelling. In the sequence to sequence translation paper Sutskever utilises "gated recurrent units" (GRUs) to build a natural language translation system which is invariant of sentence length. Here, the gated units (based on the long short term memories of [HS97]) enable the training of a recurrent neural network on the sequences of words. Without going into the detail of the training regime for the recurrent network, which is essential for sentence length invariance, it is sufficient to say that the network represents sentences in a "high dimensional concept space". Sentences are effectively reduced to a number (vector in the space), where sentences meaning similar things are grouped together. The words forming the sentences are also coded using a vector representation using the Word2Vec methodology of Mikolov [Mik+13]. On the surface, this would appear to be a technique worth pursuing in the area of meta data labelling of maps.

2.11 Relative Scales in Data Comparison

When Google Maps was first released in 2005, data visualisations were "mash-ups" of data from different sources overlayed on a map for visual comparison [Bat+10a]. Having progressed from this initial phase the aim now is to be able to make a more rigorous spatial comparison between different data sources. This inevitably leads to the question of how to compare data at different relative scales, for instance air quality data where there are around 10 London sites, with TfL bus data where there are 7,200 buses moving in London at the peak of rush hour. As the point sources of data are not co-incident, some form of interpolating data appears to be needed.

In his 1968 paper, "A two-dimensional interpolation function for irregularly-spaced data", Shepard shows examples of data interpolation taken from biology, meteorology

and geography [She68]. Distance interpolation is of critical importance to geographic visualisation, being fundamental to heat maps, contours and height maps. For a general survey of alternatives to Shepard, see [Ami02], where triangulation methods for interpolation and radial basis functions are compared. An example of Inverse Distance Weighting can be found in [Ste+04], where the technique is used to visualise carbon monoxide data from GPS tracked sensors on a 3D map of London. Alternatively, the geostatistical method of "Kriging" [Cre93] can be used for data interpolation. Krige's original paper [Kri51] was on mining and geology, while Wackernagel [Wac03] additionally cites examples of altitude mapping for temperature and modelling. In "Spatial Data Interpolation" [MM05], Mitas and Mitsova frame the spatial interpolation problem as, "...finding a function $F(r)$ which passes through the given points". They go on to say that,

> "Because there exist an infinite number of functions which fulfil this requirement, additional conditions have to be imposed, defining the character of various interpolation techniques." (Mitas, [MM05])

In the article, the authors state that "recent applications of geostatistics have deemphasised the use of Kriging", going on to suggest alternative distance interpolation functions that have gained popularity recently. Examples of Thiessen polygons and triangulation techniques for surfaces are outlined, finishing up with a group of techniques based on splines designed to maximise the smoothness of the interpolation. The question of discontinuities in the data is also discussed in relation to climate data. Weather data, as included later in this thesis, falls into this category as fronts are discontinuities in the pressure field[14]. Interpolating data across a weather front produces invalid results, much like interpolating footfall data through a building. For example, in [ACM16], a high resolution air temperature data set from sensors around Birmingham is compared to satellite information at a lower spatial scale to measure the urban heat island effect in the city. This is currently not possible to do at city scale due to the satellite resolution. Land use data is also included in the analysis, but the authors have to take into account the atmospheric stability using the Pasquill-Gifford index for the results to be valid. Their methodology is to only analyse days when the weather across the city is stable, in other words, there are no weather fronts moving across the study area. Selective data

[14]For a definition of weather fronts, see Met Office fact sheet 10 - Air Masses and weather fronts, *https://www.metoffice.gov.uk/binaries/content/assets/mohippo/pdf/library/factsheets/11_0581-airmass-factsheet-10.pdf*.

analysis would appear to be a safe methodology to adopt with this type of data.

Having researched a number of interpolation methods and types of data, the geostatistical and mathematical techniques for comparing data at different scales provides a solid foundation to work from. Where new research could add to the current body of knowledge is in the combinations of data from different disciplines. Extensive sources of data on real-time transport, weather, air quality and commuter behaviour are starting to emerge, but use of the data often requires domain expertise to acquire and interpret correctly. One avenue of research is to investigate whether this domain expertise can be coded into predictive models of the phenomena which can then be used for interpolative comparisons and analysis to look for correlations between the different systems? The idea is for the expert's domain specific knowledge to be coded into a model of space, rather than time. For this to work, though, there would need to be a model of accuracy, uncertainty or validity which fits with the interpolation methodology.

Chapter 3

Technical Background

This chapter forms the background research into the technologies required to build the MapTube website, *http://www.maptube.org*, which was created in 2008 for sharing and exploring maps online. Investigating computer architectures for geospatial computing, the motivation behind this research is that the tiled map systems behind how sites like Google Maps and OpenStreetMap display cartography are analogous to the visualisation layer in a regular Geographic Information System (GIS), such as ArcGIS or QGIS[1]. The first aim of MapTube was to make it easy for the general public to make maps from data, and so increase the amount of geospatial data in the public domain. The automatic mapping system is explained in Chapter 4, "Designing Systems", firstly in the context of mapping volunteered geographic data when the underlying data can change 60 times a second. Then, having developed a technology that can make maps automatically from data, the idea of data mining Internet data stores is explored.

With 3,439 maps currently stored on MapTube[2], there is an immediate supply of geospatial data, so an obvious question to ask is, "how can we perform spatial analysis on all the maps to extract further knowledge?". This forms the bulk of Chapter 5, "Dynamic Visualisation". Also, an Internet GIS system which can handle fast changing data is strategically well-placed to handle real-time data, a concept which forms Chapter 6. Before that, though, the remainder of this chapter explores the fundamental architectures and algorithms that are required to make this a reality.

[1] ArcGIS (*https://www.arcgis.com*) is a commercial GIS system from the Environmental Systems Research Institute (ESRI), while QGIS (*http://www.qgis.org*) is an open-source alternative.

[2] The figure of 3,439 maps is accurate as of 16 August 2017.

3.1 Hypothesis

Web-based visualisation of geospatial data has the potential to convey information to a wide audience, but it is currently bound by the limitations of the technology. Over-reliance on static data, which can be cached for performance reasons, is holding back the development of both real-time data visualisation and exploratory 'one-off' visualisation, for example the outputs of a user's model which might only be viewed once before being run again with different parameters. The potential exists for new computer architectures and algorithms utilising the 'infinite provisioning' possible with cloud storage, plus the connectivity that the Internet gives to both users and sources of data.

Firstly, with data collection, the crowd-sourcing of geospatial information is now possible on a scale not seen before. The first hypothesis is that by constructing a website, along with tools to make it easier for the general public to upload maps of data, that this will allow the collection of data that would not be possible otherwise. The second hypothesis is that the tools which make it easier to upload the data then become tools in themselves, as the computer effectively becomes the user, with the ability to discover its own data on the Internet and make its own maps. This can only come about through improved handling of geospatial data sets and the ability to interrogate Internet data stores. As an additional benefit, the automatic interpretation of geospatial data lends itself to the question of whether further knowledge about the structure of the relationships between different maps can be discovered. This is a more 'fully connected' approach to geospatial data than the current 'one data file one layer' approach of the current desktop GIS programs. New algorithms of the type just proposed will enable the handling of thousands of maps simultaneously with a scalability that is beyond the current generation of analysis tools.

The final hypothesis is that algorithms for automatically handling static data can be adapted to handle real-time streams, interpreting the streamed data automatically and unlocking further information about what is happening in city-scale systems. It is expected that new techniques for handling the mixed use of real-time and static data will also be required to achieve this.

In the chapters that follow, the first step is to develop a theory behind web-based tiled mapping systems, which allows their performance to be quantified with different computer architectures. Next, the tools for automatic data collection are developed, us-

ing experience gained while working on web-based mapping systems for volunteered geographic data. These tools are then used with an Internet data store example, automatically mapping all 2,558 variables from the 2011 Census. Once a set of data has been collected, the focus then turns to analysis of the data and map comparison algorithms. Otherwise, without these additional knowledge discovery tools, there is an argument that collecting data in this way results in nothing more than a large number of coloured maps. The value in the three hypotheses presented here is in asking questions about the relationships between the data. Finally, the real-time data problem is tackled with the automatic modelling of transport data from the London Underground used as an example. In the penultimate "Data Exploration" Chapter, the previously developed tools are brought together to demonstrate their effectiveness at tackling applied problems.

3.2 Geospatial Computing

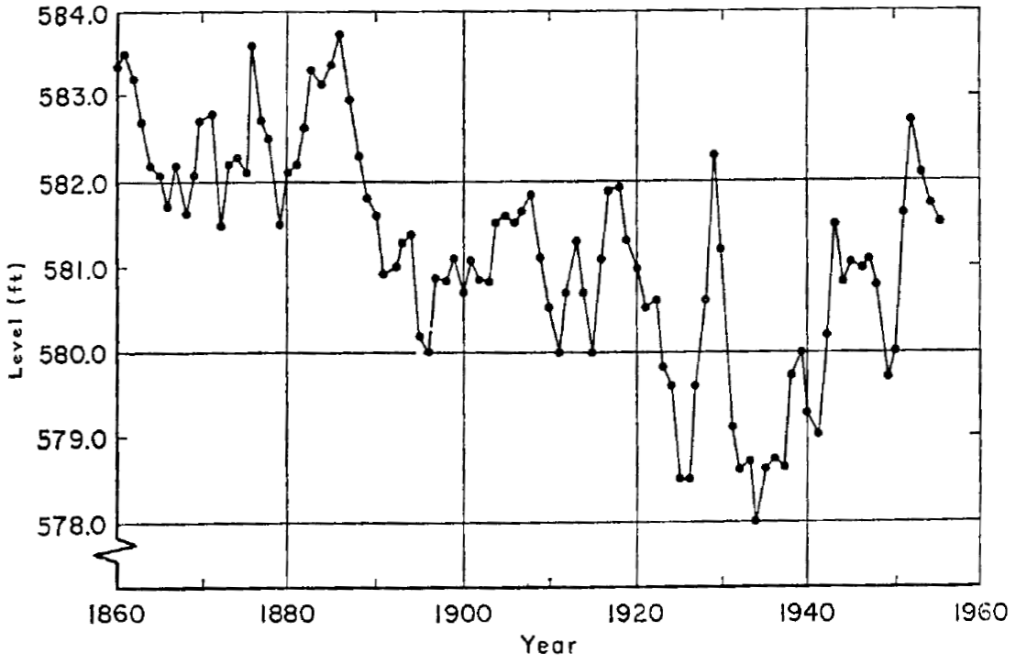
Computer Science as a discipline in its own right dates back to 1961 and Stanford University in the United States, where George Forsythe established the first 'Division of Computer Science' within the mathematics department [Knu96]. His lecture on 'The Educational Implications of the Computer Revolution' at Brown University in the same year demonstrated the growing realisation amongst researchers of the power of computers to simulate the real world:

> "Machine-held strings of binary digits can simulate a great many kinds of things, of which numbers are just one kind. For example, they can simulate automobiles on a freeway, chess pieces, electrons in a box, musical notes, Russian words, patterns on paper, human cells, electrical circuits and so on."
>
> (George Forsythe, speaking at Brown University in 1961)

Despite ESRI's claim that ArcGIS was the first modern GIS, the origins of GIS extend back to the "SYMAP" program developed at the "Harvard Laboratory for Computer Graphics and Spatial Analysis" in 1965 [Chr04]. Significant pioneering work in both computer science and geospatial computing had already been accomplished before 1961. In Fisher's 1958 paper on stratification, [Fis58], the partitioning of water level data for the Great Lakes to investigate flooding is investigated. Figure 3.1 shows his original data for lake Huron. His method was rediscovered by George Jenks in 1977 and still forms an essential part of a modern GIS system, used for partitioning data when plotting choropleth maps [Jen77]. Generally referred to as 'Jenks', or 'Natural Breaks', the technique takes a dataset and partitions it into a number of groups, while maintaining minimum deviation within groups and maximum separation between groups. It is optimal in the sense that it tests every combination, so is order $\mathcal{O}(n^2)$. Reading Fisher's original paper, he was looking at lake Michigan-Huron's highest monthly mean levels (feet) between 1860 and 1955, giving him 96 data values (K=96). Breaks were computed for 1 to 10 groups (G=10) using a "Princeton-type" computer in 3 minutes (G=10, K=96). Fisher also cites another example running on an "Illiac" computer taking 14 minutes, but recommends $K \leq 200$ and $G \leq 10$. In contrast, using a modern computer to calculate breaks for a choropleth map with 7,201 areas covering England and Wales (i.e. Census data using the Middle Layer Super Output Area geography), we would expect the Jenks breaks code to be almost instantaneous. As the dataset increases in size

though, the time increases exponentially, so, for next Census geography with data comprising 34,000 areas, the time taken to calculate the breaks has increased to minutes. The timings have been taken from the Rey paper, based on a Python implementation, [Rey+13].

FIG. 793. Lake Michigan-Huron highest monthly mean level, 1860–1955.



Source: Wallis, W. Allen and Roberts, Harry V., *Statistics: A New Approach*, Glencoe, Illinois: The Free Press, 1956, p. 567.

Figure 3.1: Fisher, American Statistical Association Journal, December 1958.

Taking this a stage further, the technique has been re-implemented in 'PySAL'[3] using parallel programming and forms part of the CyberGIS tool kit[4]. The parallel implementation of natural breaks for this tool kit, published in [Rey+13], states that the algorithm cannot be fully parallelised and Fisher's original stratification paper suggests that other solutions might be more suitable for larger datasets, leading to a possible avenue for future research.

Over the last fifty years, the original algorithm has been modified from $\mathcal{O}(n^2)$ and 200 data points, to $\mathcal{O}(k \times n \times \log n)$ (Jenks) and 7 million data points with 15 groups taking 20 seconds on a desktop PC [5], then to a parallel implementation using PySAL and PyOpenCL and 16,000 data points with a further speed-up of 1.6 over the serial

[3] PySAL is the Python Spatial Analysis library project from Arizona State University see: *https://geodacenter.asu.edu/projects/pysal*.

[4] The CyberGIS tool kit can be downloaded from: *http://cybergis.cigi.uiuc.edu/cyberGISwiki/doku.php/ct*.

[5] Source: *http://wiki.objectvision.nl/index.php/Fisher\%27s_Natural_Breaks_Classification*.

implementation. These results, obtained by Rey in [Rey+13, fig 3], show experiments with varying group sizes and dataset sizes. The GPU results using the Python OpenCL library hit a problem of GPU buffer size, so only datasets up to 4,000 points were tested. This is unfortunate, as the benchmarking results for GPU implementations of various algorithms published in Hennessy and Patterson's book, "*Computer Architecture: A Quantitative Approach*" [HP11, Ch4, fig 4.30], suggest that GPU acceleration of certain algorithms can achieve a much higher speed-up than reported here.

While the mathematical solution to the Fisher Jenks problem can be seen as a practical example of dynamic programming and optimisation, it is entirely possible that a different method exists of order $\mathcal{O}(n)$ which might be more applicable to larger datasets. Algorithms for computing quartiles using one pass over the data exist, along with approximate methods for estimating the probability density function from a data stream. Of note is the fact that, in order to parallel sort data, it is first necessary to sort the data in order to ensure that it is evenly distributed between all the computers in a cluster.

Flooding of rivers today is a big issue and serves to show how the technology has progressed. Live data at 20 second intervals can be downloaded from the NOAA website covering the Great Lakes, along with meteorology and forecast river levels derived from complex hydrological modelling systems. In the United States, the 'Great Lakes Environmental Research Laboratory (GLERL)'[6] publishes papers in hydrological engineering journals on how this prediction system works, but this is outside of the area of this thesis. An interesting history of how computing development is entwined with the development of meteorology and specifically ensemble forecasting can be found in [Lew05]. Despite the ensemble forecasting technique having been discovered by Lorenz in 1962, the first operational ensemble forecasting only started at the European Centre for Medium-term Weather Forecasting (ECMWF) in 1992. On a related note, the paper by Epstein and Parker on modelling with 6 billion agents [EP09] reinforces the link made earlier between computing power and the physical world where there are now 6 billion humans. Of special interest to this research are any problems which are currently bound by a lack of computing power.

[6] GLERL: *http://www.glerl.noaa.gov/wr/mdl_how.html*.

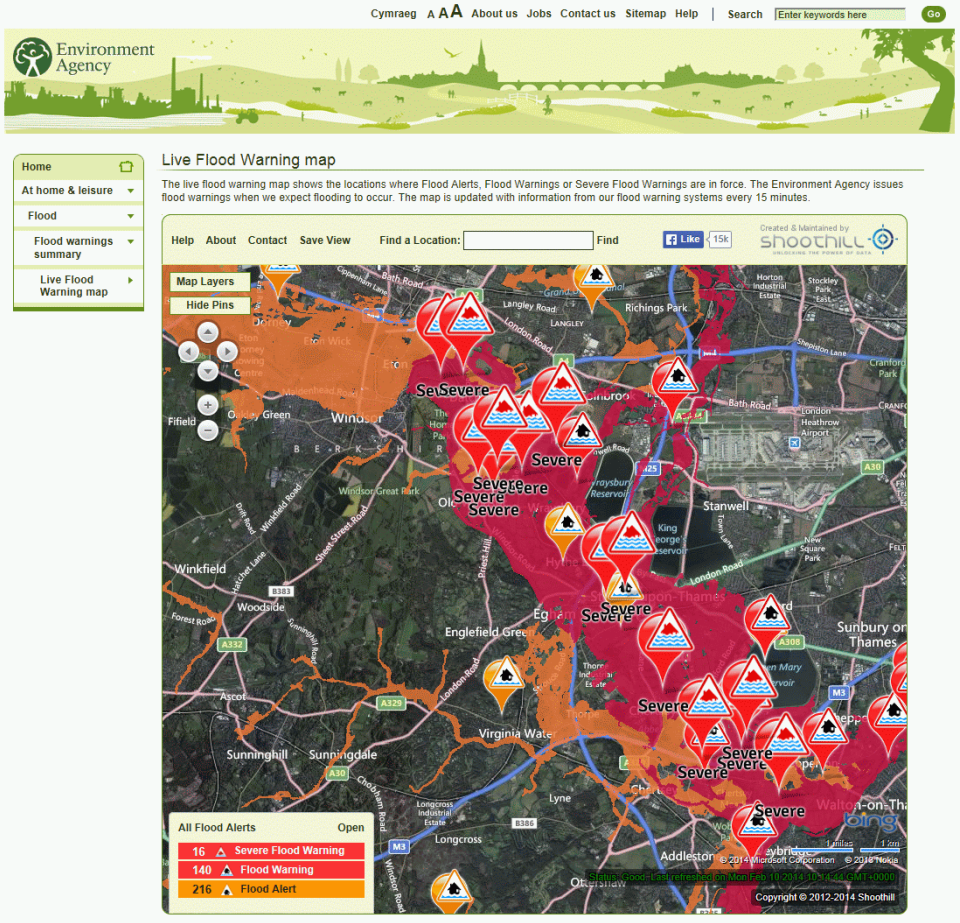


Figure 3.2: Flood warnings for the Thames around Heathrow Airport on 10 February 2014. Data copyright the Environment Agency.

Looking at the UK, the Environment Agency[7] produces flood maps updated at 15 minute intervals (figure 3.2).

These maps use data from real-time river level sensors, land height and weather forecasts to provide accurate predictions of flood risk. This is a good example of using a data fusion work flow to combine information from separate geospatial systems. The aim of this chapter is to look at the technologies behind how to achieve this type of geospatial data fusion. One other interesting point is that all this data is now real-time and in the public domain, so the opportunity exists for people to build their own forecasting systems. The 'PlanetWRF' software, available from 'Ashima Research', *http://planetwrf.com/*, is one example of open source weather forecasting, based on the NCAR and NOAA developed 'Weather Research and Forecasting' model (WRF)[8]. How accurate these open source models are when compared to the output of a national meteorological agency is something that still requires further research.

[7]The Environment Agency (EA) is the UK Government agency responsible for flood risk and hydrology.
[8]WRF web page: *http://www.wrf-model.org*.

3.3 Components of Geospatial Systems

The term "Geospatial Computing" covers a wide variety of computational tasks, relying on geometry and computer graphics for visualization and map projections to map coordinates on the Earth onto 2D projections, or 3D coordinates in the case of virtual globes like Google Earth or NASA's Worldwind. The original need for map projections was to flatten the Earth onto a piece of paper, as this was originally the only visualization medium. In contemporary systems, flat projections, virtual globes, stereopsis and animation all have a role to play, with technological advances pushing the boundaries of visualization and interaction design. Geospatial visualization for exploration and analysis of data is a major function of a geospatial computing system, with "Geographic Information Systems", or "GIS" the term used to describe this type of software. The following sections break down geospatial software systems into their component parts to enable a structured analysis.

3.4 Map Projections

Just as measurements of length and weight have must have units to be meaningful, spatial coordinates must have a well-defined mathematical link to a fixed point of reference in space, which is usually the Earth. These could be Cartesian coordinates fixed with an origin at the centre of the Earth (Earth Centred Earth Fixed or ECEF), or spherical coordinates in either degrees or radians with a mathematical model of the Earth ellipsoid. The British National Grid system is slightly more complicated, being a projected coordinate system, which is fully defined together with definitions of datum, spheroid and ellipsoid in [Ord10].

Geospatial computing systems are concerned with coordinate systems in order to draw data on the display in the correct location to a known accuracy. According to the Open Geospatial Consortium, a Coordinate Reference System is defined as follows:

> "A **coordinate reference system** is a coordinate system that has a reference to the Earth. A coordinate reference system consists of a coordinate system and a datum. Types of coordinate reference systems include: geocentric, geographic (including an ellipsoid), projected, engineering, image, vertical, temporal"
>
> "A **coordinate system** is composed of a set of coordinate axes with a known metric."

> "The **datum** defines the origin, orientation and scale of the coordinate system and ties it to the Earth, ensuring that the abstract mathematical concept "coordinate system" can be applied to the practical problem of describing positions or features on or near the Earth's surface by means of coordinates." (From OGC document 03-040, 2001-11-13, OGC 01-063r1)

The origins of map projections are rooted in two-dimensional cartography and fall into one of the following three types: Cylindrical, Conic or Azimuthal. In "The Nomenclature and Classification of Map Projections" [Lee44], Lee states that, "the projections are termed cylindric or conic because they can be regarded as developed on a cylinder or a cone", before going on to suggest dropping this terminology because, while mathematically correct, the cone or cylinder might, "bear no simple relationship to the sphere". Azimuthal projections plot rays of light onto a developable surface which forms the projection plane. The focal position of the rays defines the geometry of the projection. While Lee is concerned with the terminology of map projections and classification into groups of conformal (angle-preserving) and area-preserving, Snyder and Voxland have published "An Album of Map Projections" [SV89] for the U.S. Geological Survey, which serves as a visual illustration of the different types. In the section, "Classification based on Construction" on page 5 of this album, the types "Cylindric", "Conic" and "Azimuthal" are used as classifications for the maps which follow. While hybrids of the three types are possible (e.g. HEALPix is a hybrid of Collignon and Lambert), mapping onto general polyhedra is another alternative, for example the "Dymaxion" map of Buckminster Fuller, shown in figure 3.3d. For completeness, other projections that do not fit the three classes above are labelled as 'miscellaneous', like the Lagrange or various 'Globular' projections in [SV89]. Although the 'Proj4' library is introduced later, it is worth looking at the code behind its Java version at this point in order to more fully explain the relevance of the three types of projection. 'Proj4J' is the Java version of the open-source projection library which underpins all of the map projections used by the 'Geotools' GIS library and 'Geoserver' web map tile server. Looking at the original Java source code for the Proj4J project[9] from OSGeo, the following definitions exist for projection base classes: *AzimuthalProjection.java*, *CylindricalProjection.java* and *ConicProjection.java*. Taking the 'PlateCarreeProjection' class as an example, this extends the 'CylindricalProjection' class, which demonstrates the object

[9] proj4J on GitHub: *https://github.com/OSGeo/Proj4J/proj4j/tree/master/trunk/src/main/java/org/osgeo/proj4j/proj*.

oriented reasoning behind grouping the main projections in this way. All the concrete projection classes are derived from their respective base type, thereby re-using code and making for a logical structure which is based on the mathematics of how the projections are derived.

Map projection libraries take descriptions of coordinate reference systems in "well known text" (WKT) format, in the case of an OGC compliant library, and can derive a maths transform from one coordinate reference system to another. This is a utility library that can take data points in a projection like OSGB36 [Ord10] and reproject them into a Spherical Mercator projection for web mapping, or even convert to an unprojected spherical system like WGS84 or the Cartesian ECEF for 3D visualization. The 'Spatial Referencing' website, *www.spatialreferencing.org*, contains definitions of all the popular coordinate reference systems and is a useful reference when converting from one system to another. Where Cartesian systems like 'ECEF' are concerned, this is the difference between the definitions of "Geography" and "Geometry" in the OGC's "Simple Features for SQL" definition [Ope10b]. Geography is an unprojected spherical coordinate system, while Geometry is projected onto a flat plane. There are only two "well known text" forms for describing coordinate reference systems, either the "WKT" form for OGC derivatives or "Proj4" for the Proj4c and Proj4j derivatives. Issues arise where coordinates are being moved from one coordinate system to another where the two systems are based on different spheroids, for example, OSGB36 to Mercator, where a translate, rotate and scale operation needs to be performed as a final step.

'Proj4' is a cartographic projections library which is part of OSGeo, an organisation which publishes open geospatial standards. 'Proj4' was originally written by Gerald Evenden of the USGS and has been converted to numerous languages [Eve05]. The source code is available from *http://download.osgeo.org/proj/proj-4.8.0.tar.gz* and a quick Internet search yields a long list of ports to other languages: PROJ.4 VB (Visual Basic), fPROJ4 (MySQL), proj4rb (Ruby), pyproj (Python), PROJ4JS (Javascript), PROJ4PHP (PHP), PROJ4.J (Java), PROJ.4 (Delphi and Borland C++), Proj.net (Microsoft .net). The USGS 'Album of Map Projections' [SV89] is another useful resource which shows a number of map projections, and the mathematics behind how they work. Selected map projections are published as standards by organisations like the 'European Petroleum Survey Group' (EPSG), who govern the use of all 'EPSG' projection codes. Their website is a useful reference for looking up the definitions [Eur14]. Information

for projections usually contains the Proj4 string and the OGC's 'Well Known Text' (WKT) formats. Figure 3.3 shows a number of commonly used map projections.

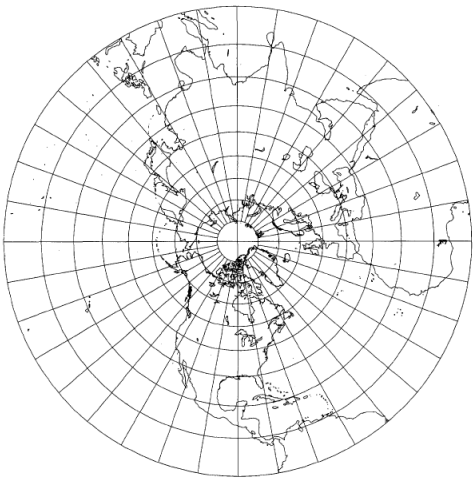
Map projection libraries form a necessary part of any geospatial system that is required to compare data in different coordinate systems. It is entirely possible to plot data without any CRS by simply scaling the viewport to fit the data extents, although this can result in the data being visibly skewed where the North and East scaling is markedly different. Although measuring a point with reference to the Earth is simple in today's GPS enabled world, it is entirely possible to collect and visualise spatial data with local, rather than global coordinates. A single fixed point from which to measure data and one, two or three orthogonal direction axes are all that is required. Aerial survey data can contain images of roads, buildings or crops where the information contained is relative to the collection area. An example of this could be image mapping using drones to determine the location and percentage coverage of an invasive species within a target area, as in [Riv+18], where the images from a downward pointing camera were stitched to form a cohesive map for analysis and geolocated later using map matching techniques. Another example is a point cloud taken of an archaeological site showing structures and artefacts with their relative positions to each other forming the spatial context, although it would be unusual not to GPS locate the instrument where this is possible. In "Simultaneous Localisation and Mapping" algorithms, or SLAM for short, 2D and 3D maps are constructed using a robot moving around the target area. The captured images are used to estimate position and pose, with the scene constructed from the relative positions of objects to each other. This technique can be used to survey indoor areas, caves and tunnels where GPS is not available.

In terms of visualisation without reference to the Earth, though, this is encountered in 3D visualisation systems where 'model world' coordinate systems have to be used. The Autodesk 3DS Max and XBox visualisations presented in this thesis are examples of this type of visualisation (figures 3.5, 3.6, 4.16 and 4.39), where data in an OSGB36 projection has been converted to model world coordinates. Here, either the size or accuracy of coordinates can cause a problem, resulting in the necessary conversion of Earth coordinates into screen coordinates. While it can be argued in this instance that there is a CRS because there is a defined mapping between the two coordinate systems, the animation does not use it and renders all the data with reference to their own world box. This is an important point to make concerning the construction of visualisa-

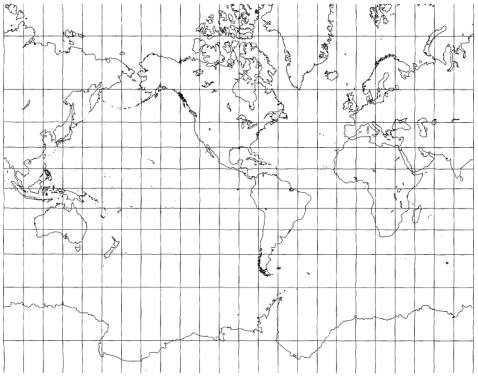
tion systems because it recognises the limitations of the computer's ability to represent numbers defining a point on the Earth. The mantissa and exponent sizes defined in the IEEE-754 specification [Ste80] result in approximately seven accurate decimal digits for single precision and sixteen for double precision. This has implications for coordinate systems like EPSG:900913, where representing the world in metres implies a range of ± 20037508. Here, double precision floating point is the only option, so a graphics library that uses single precision for speed will not have enough precision. Virtual Earths like Google Earth and NASA's Worldwind have to solve this problem to display the world, while allowing the user to zoom in close to an area of interest. However, data might only exist for a small portion of the world, for example, the London Underground maps presented later in figure 4.39, so the viewport area can be limited and the data rescaled to fit.

While most geospatial systems will probably use one of the existing libraries, it is worth pointing out that a GPU accelerated reprojection, or parallel reprojection architecture would be easy to implement and result in a large performance increase. At present, the floating point maths used will benefit from floating point unit (FPU) acceleration and 'single instruction multiple data stream' (SIMD) streaming extensions where available, but the task of reprojection is inherently mathematical and data parallel, with no sequential dependencies. Once the maths transform has been determined, it is a simple case of applying it to every point in the dataset. The only problem with this is that the world is a spheroid which wraps around from zero degrees to 360 (zero) again, so polygons which span both sides of the artificial separator will cause problems. The wrapping problem means that reprojection requires each feature[10] to be reprojected in parallel, not each individual point.

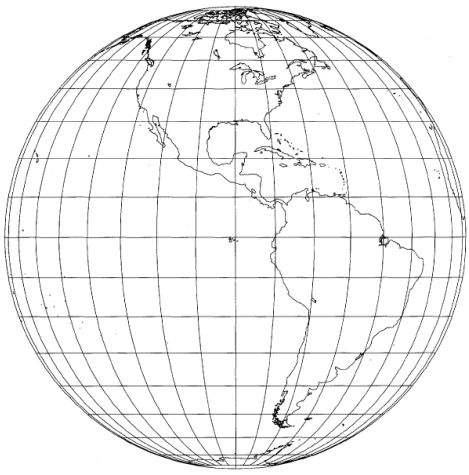
[10]Feature is defined based on the "Simple Features for SQL" definition of geospatial geometry as unique object unconnected with any other, containing a boundary that does not overlap and zero or more holes.



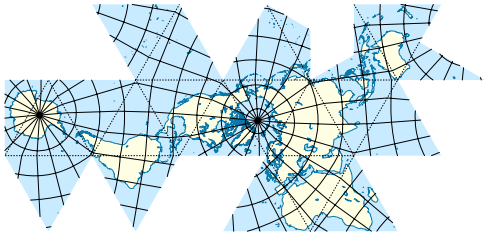
(a) Polar Stereographic Projection.



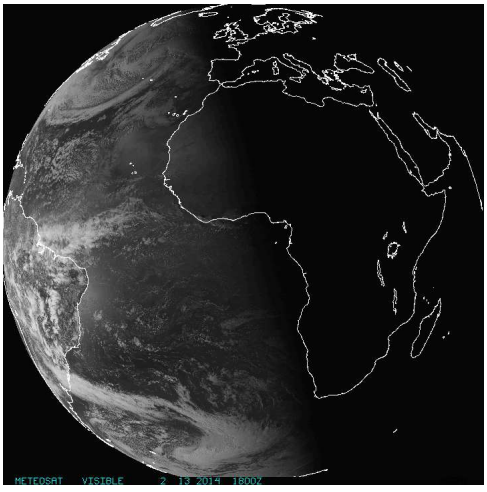
(b) Mercator Projection.



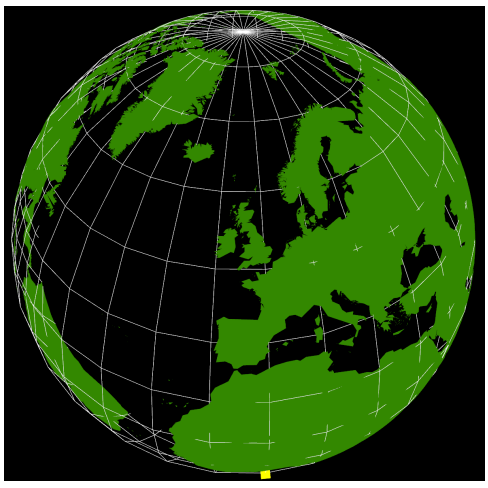
(c) Orthographic Projection.



(d) Dymaxion Projection, created by Eric Gaba (username Sting), Wikimedia Commons.



(e) Space view, Meteosat full Earth disk, visible image for 13/2/2014 18Z. Image downloaded from NOAA.



(f) Earth Centred Earth Fixed Cartesian Coordinate System. This example was rendered using WebGL in the Google Chrome Browser.

Figure 3.3: Six examples of different map projections.

3.5 Geospatial Libraries

Geospatial data can be seen as a special case of geometry, so there is considerable overlap between geospatial libraries and computer graphics libraries for manipulating geometry. Taking the Java based "Geotools" [OSG13] as an example of a widely used geospatial library[11], this uses the "Java Topology Suite" (JTS) [Viv13] for geometry, while mathematical operations for handling map projections make use of code from Proj4J. The JTS library was originally a library for performing geometric operations on general Euclidean geometry[12], not specifically for geospatial libraries. The OGC's "Simple Features for SQL" specification (SFS) [Ope10b], which JTS complies with, also includes a "spatial reference identifier", which defines how the coordinates in the geometry map onto the Earth. Where spatial databases are concerned, Microsoft's SQL Server, PostGIS and Oracle Spatial all support the SFS specification, but make the distinction between geography, based on a spherical reference, and geometry which is planar. Ultimately, though, whether geography or geometry, they are just numbers defining a point on the Earth's surface through a spatial reference. The computer is a universal computing machine, which is capable of performing the transformations required to visualise the data on the screen through buffers, graphics processors and video memory. This is the same whether the end result is a video game, flight simulator or Geographic Information System. Where geographic information is concerned, the SFS definition allows for hierarchies of objects with geometric coordinates at the bottom layer, so a complex geometry can contain objects constructed from polygons, lines or points. This is a semantic description at a higher level than the raw geometry, which can then be used to transform or query data. For example, buffering an area around a polygon region, or finding all regions which touch another and are neighbours. This is where a GIS library like Geotools sits in the code hierarchy, with computational graphics algorithms fundamental building blocks at a lower level. How all these pieces fit together to form a cohesive unit of tools is the question to be answered here. The diagram in figure 3.4 shows how all these components are built into the Geotools library.

The basic geometric functions are built on the "Java Topology Suite" (JTS) [Viv13], where algorithms originally from computer graphics are used for geographic data transformation. The map projection module is based on an Open Geospatial Consortium

[11] Geotools 9 had 14,429 downloads of the library in 7 months from January 2013 until the 10 series was released in July (Source: Sourceforge download statistics).

[12] JTS Library: *https://projects.eclipse.org/projects/locationtech.jts*

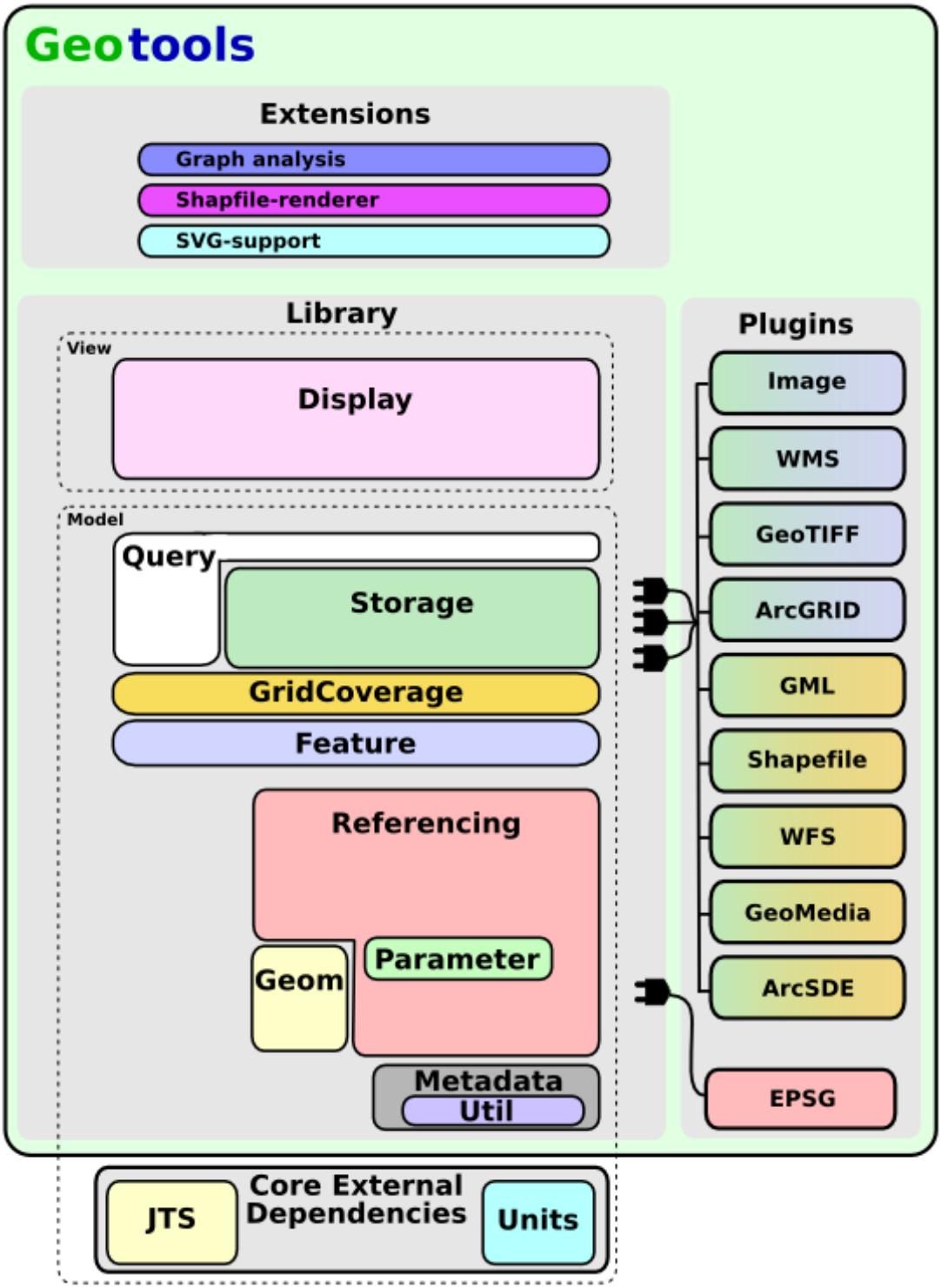


Figure 3.4: Geotools System Diagram from Refractions.net (uDig project).

(OGC) and OpenGIS compliant set of interfaces which are a set of standards to allow easy transition between geospatial systems.

In C#, Microsoft's .Net programming language, the situation is similar, with map projections handled by Proj4.net, which is another port from Proj4C and the 'Net Topology Suite', which is a port of the Java Topology Suite for C#. Most geospatial libraries implement the 'GeoAPI', so moving from one language to another is simplified by the use of a standardised set of interfaces. Although C# only ranked 8th in the GitHub language trends[13], just behind C++, the diversity of commercial .Net geospatial development can be seen from the projects on the Microsoft CodePlex Open Source site[14]. Libraries like 'SharpMap' and 'DotSpatial' build on the 'NetTopologySuite' and .Net port of the Java 'Geotools' library. Moving into C and C++, the 'GEOS' library forms a separate development tree for geometric manipulation, and is included as the base geometric library in a multitude of geospatial software. The 'Mapnik' tile renderer for OpenStreetMap is one such example[15], but the importance of C and C++ libraries is that this is often the first choice for new software, especially in the area of computer graphics. One example of this is the AutoDesk FBX library which has bindings for Python and C, and is a route for exporting geospatial data in 3D formats like FBX for 3DS Max to make professional animations, or Collada for import into WebGL and real-time visualisation. FBX and Collada are both games industry formats created by Sony for the interchange of assets between 3D modelling systems. WebGL [Khr14] is a W3C standard for HTML5 browsers to enable high performance 3D rendering using GPU acceleration for online games, but it can also be utilised for drawing geospatial data. While interoperability and linkage between different technologies can never be completely eliminated, being able to manipulate geometry in a unified way in any language is a primary requirement. As for geospatial data, the Open Geospatial Consortium's "Simple Features for SQL" standard, mentioned in the next section, goes a long way towards providing identical patterns in all languages for handling geospatial feature data.

The reason that geospatial libraries exist is that they form the basis of further development and, looking at the geospatial software currently available, there is a rich ecosystem of diverse projects. These can be categorised into layers, with the geometry

[13] Source: Language Trends on GitHub *https://github.com/blog/2047-language-trends-on-github*.
[14] For Microsoft Codeplex see: *https://archive.codeplex.com*. The site ran from 2006 until closing in 2017.
[15] The Mapnik homepage can be found at: *http://mapnik.org*.

and geospatial libraries at the bottom, followed by GIS, visualisation, analysis and web services. Key to this interoperability between systems are the standards which define how data is stored, accessed, processed and visualised.

3.6 Standards

Standardisation has occurred in recent years with organisations like the "Open Geospatial Consortium" and "OSGeo" publishing sets of standards to allow interoperability across geospatial packages.

The following quote from the Open Geospatial Consortium's website states their aim to provide standards which are used by the geospatial industry:

> "The Open Geospatial Consortium (OGC) is an international industry consortium of 475 companies, government agencies and universities participating in a consensus process to develop publicly available interface standards. OGC Standards support interoperable solutions that "geo-enable" the Web, wireless and location-based services and mainstream IT. The standards empower technology developers to make complex spatial information and services accessible and useful with all kinds of applications."
>
> (from: *http://www.opengeospatial.org/ogc*)

Recently, the OGC has published two ISO standards: ISO19111 (Geographic Information - Spatial Referencing by Coordinates) and ISO19107 (Geographic Information - Spatial Schema), which are not open standards in the sense that payment of a subscription is required to read them, so it remains to be seen whether this trend continues. Joining the OGC's mailing list or participating in their working groups to draft new standards is open to anyone. Currently, OGC web services use a combination of W3C query string requests, XML message passing or key value pairs, but there is an active debate over representational state transfer (REST) protocols, which GIS vendors like ESRI are keen to implement, that has resulted in a REST GeoServices software working group being formed (2014). The group's web page can be found at: *http://www.opengeospatial.org/projects/groups/gservrestswg*. The 'REST' standard for web services was created by Roy Fielding in his PhD thesis [Fie00]. The standard favours URL hierarchy to implement services over parameter strings. For example, "http://myserver.com/record/add" rather than "http://myserver.com?operation=add".

The Java based Geotools library is a product of the OGC, which implements new

OGC standards as they are ratified. The list of geospatial standards published by the OGC is too numerous to list here, but includes web standards for publishing maps like WMS, WMS-C, WMTS and WCS, the Web Feature Service (WFS), Web Processing Service (WPS), the data formats GML, KML and CityGML, and finally the Simple Features for SQL specification which defines geometry and geography.

The following is an outline of some of the important geospatial standards currently in use:

GeoAPI: GeoAPI 3.0 Implementation Standard A Java language application programming interface (API) which defines a set of geospatial types like envelopes (bounding boxes), features, polygons, points and lines, along with methods to manipulate them.

CityGML: OGC City Geography Markup Language (CityGML) Encoding Standard An XML mark-up for describing geographic properties of cities and the objects contained within them.

GeoSPARQL A geographic query language for RDF data, based on the SPARQL language. SPARQL is an acronym for 'SPARQL Protocol and RDF Query Language', which is a W3C standard. 'RDF' is an acronym for 'Resource Description Framework' which is another W3C standard for linked data stored as triples composed of: subject, predicate, object. These define how data items relate to each other and permit complex search graphs.

KML: Keyhole Markup Language A cut-down version of GML originally used by Google and handed back as an open standard. The 'Keyhole' reference is a legacy of Google's original acquisition of the Keyhole company which started the web-based mapping idea.

GML: OGC Geography Markup Language XML mark-up for geographic data. There is a binary encoding option and a plain text version as GML encodings of large geographic datasets can be exceptionally large. The binary encoding is a more efficient format, more in line with the size of a comparable shapefile. The ONS MSOA boundary file for England and Wales for 2011, containing 7201 geographic features, is 15.25MB as a shapefile and 38.1MB when exported as a text format GML.

CAT: OpenGIS Catalogue Service Supports the ability to publish and search collections of descriptive information (meta data) about geospatial data, services and related resources.

CTS: OpenGIS Coordinate Transformation Service Designed to provide a web-based way of performing a key element of GIS systems by reprojecting data from differing coordinate reference systems into a single projection for display.

SFS: OpenGIS Simple Feature Access, or Simple Features for SQL Defines the storage and access of geographic vector data as polygons, holes, lines and points using a well-defined interface. This is one of the most important standards of the OGC as any commercial database claiming to be 'geospatial' will almost certainly implement this standard. Examples include SQL Server 2008/12/14, PostGIS, MySQL, Oracle Spatial and many others.

SLD: OpenGIS Styled Layer Descriptor When drawing cartography, this XML markup defines how features should be displayed at the current zoom level. For example, "motorways are always drawn in blue, but 'A' and 'B' roads don't appear until higher scale zoom levels and are red". Point features can have their own graphics, so any features on an Ordnance Survey map can be rendered, which is what this standard is designed for.

SE: OpenGIS Symbology Encoding Defines symbolizers for vector and coverage data independent of WMS, which the SLD standard was designed to support. Features are styled according to rules applied to the data which control colouring, stroke width and fill.

WFS: OpenGIS Web Feature Service Defines operations on geographic features like get, query, create, search, delete or lock. The *data.gov.uk* website and the Scottish Census now offer open Census boundary files as a published WFS URI rather than the more traditional shapefile download. While this is intended to be used as a source of vector layers in desktop GIS software, formats, projections and protocol versions need to be compatible.

WMS: Web Map Service Before the advent of tiled maps, WMS was used by web clients to draw maps . A bounding box (envelope) defining the geographic area to be drawn, along with the dataset name are specified in the web request. The

server responds with an image file showing the map. This system does not scale to large numbers of requests, so has largely been replaced by WMTS. As a stop-gap solution, a variant called "WMS-C" exists where all WMS requests are aligned to fixed bounds on the server and the image to be returned is built as a mosaic out of the 'cached' tiles on the server and clipped to the requested area. This solution scales to a greater extent than plain WMS, but still cannot handle the requests made on a modern web-based mapping system.

WMTS: OpenGIS Web Map Tile Service All modern web-based mapping systems use the same projection and fixed tile architecture for performance reasons. WMTS is a retro-fitted standard designed to cover Google Maps, Open Layers and Bing Maps. It defines a quad tree method for referencing tiles where the world fits on a square and each successive zoom level doubles the number of tiles in the X and Y axis, giving a progression of $t = 4^z$ where t is the number of tiles and z is the zoom level, starting with zero as the whole world.

WPS: Web Processing Service Defines the discovery of services and their execution as processes acting on geographic data.

The GeoAPI standard[16] mentioned above is important as, despite being written as a Java interface, it provides language agnostic structures and methods for handling geospatial data. The following quote from the GeoAPI website sums up their aims:

> "GeoAPI provides a set of Java language programming interfaces for geospatial applications.
>
> The interfaces developed by the GeoAPI project include many of the data structures and manipulation methods needed for geographic information system applications. In a series of packages, GeoAPI 3.0 defines a core set of interfaces for meta-data handling, for geodetic referencing, projection and conversion. The "pending" part of GeoAPI defines interfaces for the handling of georeferenced imagery, for the construction and manipulation of vector geometry and topological data structures, and for the description and use of geospatial "feature" entities. Beyond this core, GeoAPI-pending further defines interfaces for data access and storage including sophisticated

[16] GeoAPI website: *http://www.geoapi.org/*.

filter queries, and for display."

(GeoAPI mission statement, taken from their website)

Despite the opening statement that it provides only Java interfaces, 3rd party ports of GeoAPI compatible interfaces exist in C# (GeoAPI.net), C++ (GEOS), Python (geoapi, https://github.com/ondrejsika/light-geo-api/) and other languages. 'OpenGIS' is another OGC namespace for open source interfaces to geospatial libraries and, over the last few years, much of the functionality of the Geotools Java library has been migrated to 'opengis'. Also of note is the Open Source Geospatial Foundation (OSGEO), which is another organisation that seeks to promote open source geospatial software and has helped in the development of some of the current geospatial libraries.

GeoJSON[17] is not a standard recognised by any official standards body, but nevertheless, it is a widely adopted file format for representing geographic features compatible with the OGC's "Simple Features Specification". Most modern GIS software will load and save GeoJSON files, for example, Geotools, QGIS, ArcGIS, OpenLayers, Leaflet, GeoServer, GeoDjango, GDAL, CartoDB, PostGIS and Mapnik, with additional support for GeoJSON from Bing Maps, Yahoo! and Google via import and export APIs. The format is text based due to being based on the "Javascript Object Notation" data-interchange format,[18] which is used in web programming to pass data to the Javascript code running on the page. GeoJSON allows geographic data to be encoded in a format which is efficient for the Javascript engine to parse, but its plain text based format can make for very large files when encoding complex geographic data. The standard technique is to 'gzip' the data to reduce the download time for the web page. An alternative solution exists in the form of the "TopoJSON" format[19] which is similar to GeoJSON, except for using a more efficient encoding for points which eliminates duplicates and reduces the file size. Protocol Buffers, which is a format introduced by Google, are another way of encoding key value pairs and also geographic data, which is currently used in the MapBox GL system for encoding vector tiles.

3.7 Procedural Generation

The procedural generation of graphics from simple rules is described in [Fol+93, Ch20] and by Smith, in [Smi84], as "Database Amplification" as it was originally used to add

[17] See: *http://geojson.org/*.
[18] See: *http://json.org/*.
[19] See: *https://github.com/mbostock/topojson/wiki*.

content and increase the visual realism of scenes where graphics created manually by artists would be too time-consuming. Alvy Ray Smith, working for Industrial Light and Magic in the 1980s, details some of the computer graphics special effects which were first pioneered in the film industry, but have since moved into mainstream use. Realistic fractal plants and mountains are explained in [Smi84], using the Lindenmayer grammar system to control their creation, while in [Smi82], the techniques used to create an entire fractal planet are explained, along with the first use of a particle system for the fire which engulfs the planet. Ken Perlin's work on Perlin noise in 1982 [Per85], was also inspired by his work in the film industry, while attempting to create realistic looking natural textures. More recently, the procedural generation of cities starting from real-life land use maps or transport networks has become possible. Kelly and McCabe [KM06] include a comprehensive survey of techniques used in the procedural generation of cities, including L-system grammars, the 'TerraGen' terrain generation program and procedural techniques for creating realistic buildings. Their follow-up paper a year later detailing their development of the 'CityGen' software as a development of Kelly's masters thesis [Kel08], completed the year after, shows the methodology behind an entire system for creating a virtual city. More recently, the release of ESRI's 'CityEngine' in 2013[20], which resulted from their acquisition of the company, "Procedural Inc.", earlier that year, brings the generation of realistic 3D virtual cities into the GIS domain. The use of 'CityEngine' to implement a classical 'von-Thunen' model is detailed in [Rou13].

3.8 Computing Architectures

This section is split into sections on: general purpose programming using graphics processing units, cloud computing, parallel and heterogeneous computing, 'cyberGIS' and 'big data'. The aim is to provide a general overview of the architectures available as a prelude to the next chapter. Where more detailed theoretical analyses are required for design purposes, the information contained in [HP11] has proved invaluable. The 'single' and 'multiple' instruction stream models (SIMD and MIMD) refer to Flynn's definition in "Very high-speed computing systems" [Fly66].

[20] See: *http://www.esri.com/esri-news/releases/13-4qtr/esri-cityengine-2013-brings-powerful-modeling-to-your-favorite-3d-applications*.

3.8.1 General Purpose GPU

For high performance computing on the desktop, General Purpose programming using Graphics Processing Units, or GPGPU programming, has become a mainstream technology that demands serious consideration. High numbers of SIMD processing cores are available to any algorithm that can make use of them. The AMD "Aparapi" project [AMD11] is an interesting mid-ground between full GPGPU programming and conventional software. By intercepting Java byte code as it executes on the Java Virtual Machine and executing on the GPU instead, Java code can be accelerated without modification. However, it can only execute the instructions that it is given. What it is not able to do is find an improved, or parallel, solution to a problem, so for example, it won't find the butterfly algorithm for FFT (Fast Fourier Transform) when given the DFT (Discrete Fourier Transform) code [IJ93]. An obvious case for GPU acceleration is reprojection, as computer graphics and spatial coordinate transformations have the same mathematical origins. Another example comes from data mining where the GPU has been used to accelerate the k-means algorithm [Fan+08]. This is an example of an algorithm that is very similar to a matrix multiplication, but with the loop altered slightly, so that it maps very easily to the SIMD GPU. One interesting omission from the Fang paper is the absence of a parallel mean implementation to find the new cluster centres after the initial cluster distance calculations. The chosen technique is to move the data back to the CPU and perform the calculation there. This demonstrates the biggest problem with GPGPU programming, namely getting the data onto the GPU, which only has comparatively small buffers, and then transferring the results back to the host processor. This is where GPGPU programming is described as heterogeneous rather than pure SIMD. Performance tests in [Gas12] show the effects of using different types of buffers and how this has a significant effect on the execution time.

3.8.2 Cloud Computing

The term "Cloud Computing" refers to an architecture which spreads processing and data across many servers, storing data in multiple redundant locations and so offering a level of resilience that is now expected from the major Internet companies. These systems are built to be big and scalable, running on cheap commodity hardware with fault tolerance emerging from data replication. Central to how this architecture functions is the MapReduce algorithm [DG08] which was published by Google in 2008. Many vendors offer this type of cloud service for hire at a cost per compute cycle, or bit of

data stored. Examples include Microsoft's Azure platform, Amazon's compute cloud and Google's AppEngine, or Hadoop, Casandra (Facebook), HyperTable and a host of other 3rd party projects. The basic architecture uses a distributed file system, for example HDFS, with a compute layer running on top which allows queries to be made. Although this is a very simplistic view, it makes the important point about the ability to analyse large datasets and harness more compute power than has previously been available to the general public. It also shows the move away from relational databases and towards the analysis of unstructured data which does not fit easily into the entity relational model. In addition to this, GIS systems like Geotools are beginning to be seen running on these distributed architectures.

It is worth pointing out that, while the web as we know it wouldn't be possible without cloud computing, it is not without problems as the following two quotes explain:

> "The interesting thing about cloud computing is that we've redefined cloud computing to include everything that we already do. I can't think of anything that isn't cloud computing with all of these announcements. The computer industry is the only industry that is more fashion-driven than women's fashion. Maybe I'm an idiot, but I have no idea what anyone is talking about. What is it? It's complete gibberish. It's insane. When is this idiocy going to stop?" (Larry Ellison, CEO Oracle[21])

> "One reason you should not use web applications to do your computing is that you lose control. It's just as bad as using a proprietary program. Do your own computing on your own computer with your copy of a freedom-respecting program. If you use a proprietary program or somebody else's web server, you're defenceless. You're putty in the hands of whoever developed that software." (Richard Stallman, GNU FSF[22])

To put these two quotes into context, Larry Ellison, as the CEO of Oracle, has an interest in distributed databases rather than Internet technologies and his quote follows on from Oracle 9G using GRID technology. The later acquisition of Sun Microsystems and Java and the release of Apache Hadoop tools for the core database show the fluid way that the company positions itself in the market. Richard Stallman makes a good

[21] Taken from Larry Ellison's keynote speech at OpenWorld in September 2012 *http://www.oracle.com/openworld/keynotes/index.html* (February 2013).

[22] Quote from Richard Stallman of the Free Software Foundation in a Guardian interview in September 2008 *http://www.guardian.co.uk/technology/2008/sep/29/cloud.computing.richard.stallman* (February 2013).

argument for maintaining control over ownership of your own data and the inability to look inside the box and see how a cloud computing application works.

Where cloud computing fails is when large amounts of data need to be moved between the cloud and client machines. In this situation, the Internet connection is the weakest link, limiting the transfer speed. At the time of writing, 100MBit connections are typical in Universities, with this slowly being replaced by 1GBit, but only due to the direct connection. More typical of home users is 12MBits, with some companies providing 50MBit where a fibre connection is available. In general, the way this is handled is to ensure that the processing of large or real-time data occurs on the same system as the one holding the original data. The Network Rail API for all UK train movements is an example of this, being hosted on the Amazon cloud. This fact is advertised in the documentation and clients are advised to host their own services on the Amazon cloud to take advantage of the faster transfer speeds. Another example is the TfL Trackernet API for London Underground, which is now hosted on Microsoft Azure after originally launching on TfL's own in-house servers and failing under high load. Best practice when trying to track all public transport in London when the data is split between Amazon and Azure is still not fully investigated. This could be the Achilles' heel of cloud computing with large and fast datasets.

Moves are already being made towards cloud-based GIS systems, both in the commercial market and open-source communities. The release of 'ESRI's GIS Tools for Hadoop' (ESRI, 2013) on 25 March 2013 shows how they are positioning themselves in the market, but the indicators have been around for the last year at least. Reading the blog of Mansour Raad[23], a senior software architect at ESRI and Cloudera Certified Developer for Apache Hadoop, there are examples using the core Hadoop project and its other closely associated Apache projects of 'Hive' [Apa16a] and 'Pig' [Apa16b]. The Apache 'Hive' project deals with the storage of relational data on the NoSQL HDFS file system which Hadoop uses, while 'Pig' is the Swiss army knife of real world data processing on Hadoop. Pig provides job control for MapReduce jobs run on Hadoop. Because of the way the functional MapReduce programming works, a typical task will be split into a number of dependent jobs, so, for example, the data needs to be moved from the local file system onto the remote HDFS, then one or more parallel map and reduce iterations will be used to transform the data, using collation and sorting where

[23] Mansour Raad's blog: *http://thunderheadxpler.blogspot.co.uk*.

necessary. Then the task will finish by collecting up all the outputs and writing the result back to the local file system. For a full description of MapReduce and distributed file systems see [DG08].

Where real-time processing with MapReduce is required, the "Spark" [Apa16c], "Storm" [Apa14a] and "Shark"[24] projects adapt the original job-based methodology to a variation that runs in memory with a more immediate turnaround time for jobs. These projects are all interesting in that they are second generation Hadoop developments which are spin-offs from attempts to adapt the technology using real world experiences. In a similar thread, data mining and machine learning are significant areas of interest in "Big Data Analytics" and one that is being addressed by projects such as "OpenML" [Rij+13] for machine learning and "Massive Online Analysis" [Uni14] for data stream mining. These projects are a potential developmental next step in spatial analytics and real-time spatial processing if the techniques and experience are transferable to the spatial analysis domain.

The following quote about the applicability of parallel cloud computing techniques is relevant though:

> "MapReduce is not always the best algorithm. MapReduce is a profound idea: taking a simple functional programming operation and applying it, in parallel, to gigabytes or terabytes of data. But there is a price. For that parallelism, you need to have each MR operation independent from all others. If you need to know everything that has gone before, you have a problem."
>
> (Apache Hadoop Wiki: *http://wiki.apache.org/hadoop/HadoopIsNot*)

In his masters thesis, Nathan Kerr [Ker09] analyses the performance of a "Spatial Hadoop" system and compares it against an alternative system built using message passing. While the spatial Hadoop system comes second in this performance test, utility computing is now so widespread and cheap to use that its value cannot be underestimated. A full description of utility and warehouse scale computing can be found in Chapter 6 of [HP11], where the economies of scale and use of cheap unreliable hardware that make today's cloud computing possible are analysed. Much of the information behind this chapter comes from a release of information by Google on their shipping container computers in 2006, so it is generally assumed that the current state

[24] Originally a separate project, Shark is now an Apache Spark module called Spark SQL. The original Berkeley project is archived at: *http://shark.cs.berkeley.edu*.

of the art is a long way ahead of this. Conversations with current Google employees would seem to support this assumption. The Apache "MESOS" project [Apa14b] comes from the "The Datacenter Needs an Operating System" work of Zaharia et al. [Zah+11] where they highlight the need for effective job scheduling. At the scale of a warehouse computer, even a 1% improvement in efficiency is significant. The work of the Berkeley AMP Lab has led to these computing architectures becoming mainstream as the technology has filtered down from the cloud computing providers where it has been tried and tested and proved to work. The main attraction of this utility computing is its 'Pay as You Go' nature, otherwise termed 'Infinite Provisioning'. This means that, for a piece of analytical work requiring more computing power than is available to an organisation, it is cost-effective to pay for only the compute power required to accomplish the task.[25]

3.8.3 Parallel Algorithms and Heterogeneous Computing

There is a distinction to be made here between high performance computing and scalability to high numbers of requests. Web sites that have high numbers of users load-balance requests across servers, which is a task parallel operation as each thread is an individual thread of execution with no dependency. General high performance computing is a more difficult problem due to MapReduce not being the best paradigm and the non-vectorizable elements of the code. This fact is stated clearly on the Hadoop website[26]. Amdahl's law is rarely cited outside of supercomputing, but illustrates the diminishing returns as the number of processing elements increases:

$$Speedup_{overall} = \frac{1}{(1 - fraction_{enhanced}) + \frac{fraction_{enhanced}}{speedup_{expected}}}$$

(Amdahl's Law, reproduced from [PH90])

Here, $speedup_{expected}$ is the number of parallel processing elements. Four cores implies a four times speed-up. The key is the $fraction_{enhanced}$ term, where $1 - fraction_{enhanced}$ is the part that can't be vectorized, for example a sequential dependency. As the number of processing elements tends to infinity, then:

[25] An interesting point to note is that early supercomputing often followed this time-share computing model. An often quoted fact is that the UK Meteorological Office's first forays into numerical weather prediction were when they were based in Kensington in the 1950s and rented time on a Leo computer which had been developed by Lyons, the company behind Lyons tea rooms, to run their payroll [Inn10].

[26] Hadoop 'What Hadoop is Not' explanation: *https://wiki.apache.org/hadoop/HadoopIsNot*.

$$Speedup_{overall} = \frac{1}{1 - fraction_{enhanced}}$$

This result is important as the Teraflop figures often quoted for GPUs are based on 100% utilisation, which is very rarely the case in practice. The advantages gained by using GPGPU programming are through use of the large number of processing elements available. Performance gains are most likely on data parallel tasks, of which, data mining is perhaps the most relevant example (see [Wen08]) with decision trees, association rules and clustering algorithms. In the geospatial domain, reprojection can gain from parallelism, along with many stages of the visualisation pipeline as graphics and matrix manipulation is what GPUs were created for. Other algorithms like natural breaks and spatial similarity measures can also benefit with some modifications, so the beginnings of a scalable framework for geocomputation start to emerge.

While on the topic of parallel architectures and vectorization, it should also be pointed out that any task parallel applications are also an obvious candidate for inclusion. Ensemble forecasting falls into this category where the same model is run many times with slightly different input conditions to determine how sensitive the result is to the input. This type of analysis is used to generate confidence maps, for example in weather forecasting where the technique was first used in the 1950s for ensemble forecasting after the dependence of numerical weather prediction models on the starting conditions was recognised [Lew05]. The emergence of numerical weather prediction is tightly coupled to the emergence of computing architectures capable of processing the required number of calculations, as is demonstrated by diagram A1 in [Lew05], which shows where digital computation fits into the bigger picture. It also serves to show that, while the ensemble forecasting technique was proposed in the 1950s, it took several decades until the 1990s for computing power to increase to the point where routine ensemble forecasts were being produced by the European Centre for Medium-term Weather Forecasting (ECMWF). The remit of the ECMWF is to facilitate joint European research into new weather forecasting theory and techniques, which eventually mature into the national operational weather forecasting models.

One final point on GPU architectures relates to the fact that it is mainly a client side architecture due to the fact that most servers don't have high performance GPU hardware. The definition of a server is that the Operating System is configured to favour background processes over foreground ones and a physical console is rarely used on a

server, so advanced GPUs are an unnecessary expense. Most servers use a reference Matrox G200 GPU, which has been the industry standard for decades. There are exceptions to this, as Microsoft Hyper-V introduces a software emulated GPU for VMs which, although very slow, can be used for simulation purposes. The other exception is the NVIDIA Tesla and Kepler range of servers which are purpose-built GPU clusters offering very large numbers of processing elements for custom written software. These have been used very successfully for computational fluid dynamics and visualisation, but are expensive. As an example, an NVIDIA Tesla C2075 card launched in November 2011 originally cost $2,500 (US) new, but can now be bought new for $1,800. The recently released (July 2013) NVIDIA K20X currently ships for $3,800 (US)[27]. The C2075 has 448 thread processors, while the K20X has 2,688, which demonstrates 20 months of progress in the GPU market. NVIDIA Tesla and Kepler rack mounted servers based around these GPUs generally contain between 4 and 8 GPUs and cost in the region of $15,000 (US), although buying last year's top of the range processor is the more economic option. As this technology becomes more widely used, it can be expected to reduce in price, just as the games market targeted GPUs have reduced in price with mass market appeal.

3.8.4 CyberGIS

The term "CyberGIS" has been used by Shaowen Wang of the National Center for Supercomputing Applications (NCSA) in various papers and books [Wan10] [Che+13]. The idea is interesting in the way that a geocomputation system is split up into a number of discrete functional elements which are implemented as web services. The prefix 'cyber' in CyberGIS is a reference to cyber-infrastructure, which is used in the U.S. to define computing systems which are distributed across the Internet and which allow cross-discipline collaboration between institutions. This definition comes from [MGH13], along with an example which analyses London's transport network using real-time data. Here, infrastructure in the form of a real-time data server has been built which supplies data to another server designed for analytics. One barrier to the adoption of this type of system is that no established standard for geospatial operations exists. While the OGC's "Web Processing Service" (WPS) defines a standard for querying the capabilities of a web service endpoint offering geocomputation services which are open for anybody to use, what it doesn't define are the operations themselves.

[27] Archive prices from Amazon.com using the camelcamelcamel.com price tracker website.

In other words, no standardization exists for advertising the functions themselves and consequently, there is no means of inter-operability between different organizations.

In his article on CyberGIS [Wan10], Shaowen Wang outlines the paradigm of using the "software as a service" model to offer GIS functions via a web service. In essence, this can be seen as the breaking up of a traditional GIS application into its component parts and allowing users to create their own work flows from data acquisition, cleaning and processing through to final visualisation. This also opens the door to a scalability in geoprocessing systems that could potentially mirror the revolution that MapReduce had to the web. In some respects this can be seen as a natural progression as the tile renderers for web mapping services like Mapnik[28] and CASA's own 'MapTubeD' system[29] already perform the visualisation function which is the top level of a GIS system. Looking at the Open Geospatial Consortium's list of standards, the Web Processing Standard [Ope07] is intended to allow for inter-operability between systems offering GIS type functions via a web service. The only problem with this is that there is no standard for the operations themselves, no semantic description of the GIS operations and no minimum agreed set of GIS operations that a CyberGIS system should implement.

Despite some obvious problems of data bandwidth, the idea of splitting up a regular GIS into its component parts is an attractive proposition. This follows the current software paradigm of smaller discrete pieces of software with specific tasks which are more maintainable than larger pieces of monolithic code. Extensibility and customisation are also good reasons for structuring a project like this as it fits the open source methodology whereby people can fork, develop and merge new features in an ad-hoc fashion. The real key to this working is in the standards to connect geospatial systems together which are lacking in some areas.

3.8.5 Geocomputation in the Internet Age

Despite having origins that go right back to DARPANET and ARPANET[30] in 1969, the Internet of 2013 bears little resemblance to its predecessor. Visualisations of all kinds of data are now an everyday part of modern life, whether in the form of an App for a mobile phone or tablet computer, a web page on the Internet or a segment on the television news. Intrinsically linked to this data-rich society which we now live in is an

[28] Mapnik: *http://mapnik.org/*.
[29] MapTubeD is a dynamic tile rendering system introduced in section 4.1.
[30] See: *http://www.computerhistory.org/internet_history*.

underlying infrastructure to capture and deliver this information to us. The Internet is now mobile and able to both deliver data and be a source of information in itself. GPS in phones means that anything we do can be geolocated, opening the door to a greater quantity of machine generated data.

Google's search business, which is built around their search engine and indexes all the web pages on the Internet, relies on a parallel architecture using redundant computers. In the book, "The Datacenter as a Computer" [BH09], Barroso outlines the architecture developed by Google and termed 'warehouse scale computing'. This is also covered in [HP11], which gives an insight into the economics of computing at this scale. While Google's business was built around their own 'BigTable' and 'Google File System' (GFS), open source competitors developed an architecture along the same lines, called 'Hadoop'. The key algorithm under-pinning both systems, though, is a technique termed, 'Map-Reduce'. The 'Map' phase aims to spread tasks across computers in a cluster for processing, after which the 'Reduce' phase brings all the results back together. As an example, if the task is to count the number of words in a book, then spread the pages of the book evenly across processors and allow the count to occur in parallel, then add up all the counts in the reduce.

Terms like "Big Data", "Broad Data" and "Fast Data" [Slo12] highlight the different challenges that are now emerging from the data available to us. While "Broad Data" might not be intrinsically big, it is highly cross-referenced and linked, so its value is in its combination with similar datasets. "Fast Data" is not a new phenomena, but describes the real-time storage and processing of data. Fast in this context could be defined with respect to the processing algorithms applied to the data. In other words, it is no use being able to forecast weather two hours ahead if it takes the algorithm two hours to run. It is these two concepts which appear to be more interesting when combined with data stores and real-time APIs later in the section. The current trends in "Open Data" and "Open Government" mean that there is now a rich fabric of interconnected data in the public domain that has yet to be fully exploited. Public APIs, which give us access to previously unseen facets of city systems, are available due to the pervasive automation of the modern world. Information systems are ubiquitous and control a diverse range of smart city systems, from mass transit to finance, retail, hydrology and security [Keh+11]. The challenge here is to use the empirical data to discover how these systems interact, and the key to this is in building the next generation of tools

which can handle the new data.

The buzzword of 2012 was, arguably, "Big Data", but this is a concept that is notoriously hard to pin down with a simple definition. One example is as follows:

> "Big data is data that exceeds the processing capacity of conventional database systems. The data is too big, moves too fast, or doesn't fit the strictures of your database architectures."
>
> (Edd Dumbill, O'Reilly Big Data Now [Slo12])

While this definition from the O'Reilly book "Big Data Now" distances Big Data away from the fixed schema of the Relational Database Management System (RDBMS) first proposed by Codd [Cod72], the idea that database systems are 'conventional' or not scalable to big data sizes does not really fit. Oracle 10, 11 and 12G databases are scalable across a computing grid and their Exadata hardware has been running large databases (40TB per cabinet) on bare metal for many years.

The quote is also circular in nature by defining Big Data in terms of the tools that are needed to process it and not in terms of the data itself. An alternative quote demonstrates the circularity problem:

> "Big Data is when the data is too big to be manipulated on a single computer."
>
> (John Rauser[31])

This definition opens the door to the sort of distributed computing found in MapReduce and Hadoop, but does at least give some notion of data that is bigger than a single desktop machine can handle.

Finally, this is the most succinct definition, even if it isn't really a definition at all:

> "Big Data is defined as when the size of the data becomes a part of the problem." (Mike Loukides [Lou10])

In "What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets" [KM16], Kitchen and McArdle begin with three key Big Data traits of "volume", "velocity" and "variety" ("the three Vs"), before analysing seven further qualities of 26 datasets. They come to the conclusion that, "there are multiple forms of Big Data", and that "the 3Vs meme is false and misleading". Looking at the problem from the other end, in "BigTable: A Distributed Storage System for Structured Data"

[31] Speaking at the Boston Big Data and High Performance Computing Summit hosted by Amazon Web Services in 2012.

[Cha+06], the authors analyse the performance of the infrastructure behind Google's "BigTable" and "GFS" technologies, showing "MapReduce" jobs running on hundreds of tablet servers. "MapReduce" is defined in "MapReduce: simplified data processing on large clusters" [DG08], which is the algorithm used to split (map) large jobs across multiple warehouse servers and then reduce the results back down for analysis. Once this idea was in the public domain, it was re-engineered and released as open-source in the form of the Apache Hadoop project. This is the substrate that Big Data is stored on and where Big Data jobs are run. In "The Real-time city? Big Data and smart urbanism" [Kit14], Kitchen writes about the smart city and real-time city, commenting on the CASA "London City Dashboard"[32]. The real-time transport data on the London City Dashboard is directly dependent on the work in the real-time section of this thesis and the "Adaptive Networks for complex Transport Systems" project. The live system is running on the infrastructure described in section 4.5, "Real-time and Programmable Maps", which is the basis for the further work in chapter 6. This is published as a chapter in the "CyberGIS" book, "CyberGIS: Fostering a New Wave of Discovery and Innovation." [Che+14].

While these attempts at definitions all focus on data as information dumps, there is a case for defining Big Data in terms of the algorithms that are being run on it. If we consider the case, as is often cited in Geography, that all the cartographic data for the whole world constitutes big data, then geographers have been doing big data for a long time. The important point is that running algorithms against the whole dataset is extremely rare. While you can ask questions like, "where is the tallest building?", because of the nature of space, the data is partitioned into grid squares or cities. Even companies like Google and Microsoft, who have complex work flows to take Lidar data, textures and GPS data to automatically make 3D models, rarely work on larger chunks than discrete spatial blocks. This is more of a question of hierarchy and efficient spatial indexing.

The fundamental question to be asked of "Big Data" today is whether it is an enabling technology that allows discoveries to be made that were not possible before, or whether it is just a scaling effect. In the O'Reilly Strata book, "Big Data Now" [Slo12], the existence of a Moore's Law for data is suggested, based on the increase in density of disk storage over many decades. Using these long-term stores of data, it is possible

[32] City Dahsboard: *http://citydashboard.org*.

to look for infrequent events or long term events and hopefully discover models to predict them. Climate change is an obvious example of the use of very long-term data, but other applications like detecting road traffic congestion from real-time GPS of cars on the road or predicting crime hot-spots from archive data[33] are already in use today.

3.8.6 Websites for Web-Based Mapping

Google launched their 'Fusion Tables' website in June 2009, following CASA's MapTube release by two years. In addition to this, ESRI released their own ArcGIS online system with the ArcGIS 10 release in 2010. Both of these systems contain functionality to allow users to upload their own geospatial data with minimum effort, with users 'owning' a set of maps that they can choose to share with others. These systems work by having users upload data to either the Google or ESRI cloud data store. While this works for data that can be published once and viewed multiple times by different users, it is not a system suitable for exploratory GIS where data might only be viewed once and then changed. Imagine an online spatial interaction model, for example the QUANT website: *http://quant.casa.ucl.ac.uk*, where users can run employment or transport scenarios for the U.K. based on the 2010 Census travel to work data. A user can input data for a scenario, for example their own employment pattern based on the Heathrow extension being built, then run the model and visualise the results. Model results of this type are peculiar to the user who ran the scenario, though, and are not viewable by anybody else until that user decides to publish the map for general public consumption. This might only be following multiple failed scenario runs where the data is viewed once by the user and discarded as wrong for some reason. This fine-tuning process can go on for as long as required, until the urban modeller is satisfied with the output and decides to make it available for everyone, at which point it is published and could easily be viewed by millions of people. This is the distinction between exploratory GIS and publishing, because the computer architectures behind them are different. In the publishing scenario, caching is an option for load balancing requests, but in the exploratory GIS example, no caching is possible as the data is always changing, so requests need to be satisfied by the application servers.

Sites like Fusion Table and ArcGIS Online are useful in that they provide easy access to the same type of open geospatial data that is behind MapTube's ability to make maps, like Census boundary data, countries of the world and postcodes. The smaller data

[33] See Jeff Brantingham's UCLA pages on the PredPol application currently in use by the LAPD: *http://www.predpol.net*.

sets can be handled by websites using KML or GeoJSON, which is loaded in a single block and displayed as an overlay on the web page using either SVG or WebGL. Using WebGL as the rendering system, data containing millions of points can be handled, but this is not enough to cope with the finer scale Census data which is exactly the type of rich geospatial information likely to be of the most interest. To get around this problem, data is usually shown at an appropriate level of detail for the current level of zoom, which is where the map tiling technologies play their part with hierarchical level of detail. Having run the MapTube website and collected exactly this sort of data for a number of years, the challenge is in how the data can be used and interpreted by researchers. While both Fusion tables and ArcGIS Online provide some limited capability above basic choropleth maps, this stops short of the more advanced functions found in a desktop GIS that can provide real insight into the nature of the data, and neither provide any functionality to make comparisons between maps created from different sources.[34] This could be legal restriction, as it is entirely possible that Google and ESRI are worried about users adding value to data where the rights are owned by somebody else, and then re-publishing it. In this event, it would be hard to track intellectual property rights, but the fact remains that neither Fusion Tables, nor ArcGIS Online provide a set of features to allow the exploration of the relationships between the data in the different maps. Considering that these are both large Internet data stores containing geospatial data, the ability to analyse that data and place it into context seems like an omission. As a result of Internet data stores, for the first time, a wide variety of geospatial data is now available at scale, so this is the first time that anybody has been able to ask the question of how all the data sets relate to one another. This is not something that can be answered using the simple one data set model of a typical desktop GIS.

3.9 Programmable Maps

Previous sections have touched on real-time maps, but without exploring how to handle data that is constantly changing. Real time is likely to be coupled with web-based as this is the natural medium for outreach to the general public and real-time visualisations are likely to be conveying information of use to people now. Taking a transport scenario as an example, maps can show which tube lines are suspended and which stations have the

[34] To qualify this statement further, users were allowed to 'mash' their own maps and manipulate point data to some extent, which was a feature added to ArcGIS online, but wholesale comparisons between assets owned by different users is not possible on either system.

most congestion so that commuters can avoid those places and not make a bad situation worse. Weather is another obvious example with maps being constantly updated with new forecasts and observations as they arrive.

Maps of this type can be seen as a work flow defining how to automatically go from raw data through to geospatial visualisation. In order to do this, the fundamental operations outlined previously need to be chained together programmatically in the correct order. Taking this a step further, it is easy to imagine a command line GIS of a type similar to GRASS [Ope16], but along the lines of the jsFiddle website (*https://jsfiddle.net*) for developing Javascript. In this case, geospatial operations on data can be defined, resulting in a web based map with the result turned into a real-time data work flow.

When the data is animated, as is the situation with tubes, trains and buses, the visualisation needs to show movement, but with data updates occurring perhaps only once every few minutes. In this situation, the positions are being forecast into the near future (now-casting) based on the last available data until the next data frame becomes available. One method for achieving this is to see the vehicles as agents and to build a simulation using agent based modelling. Building a simulation of this nature, where tube agents are programmed with a realistic behaviour derived from the real world tube network, has the secondary effect of allowing experimentation on a real city with real data. Figure 3.5 shows a proof of concept using the 'AgentScript' Javascript library to run a test model on top of Google Maps[35].

[35] AgentScript is an open-source agent based modelling framework for javascript inspired by NetLogo and developed by Redfish. It is available for download at: *https://github.com/backspaces/agentscript*.

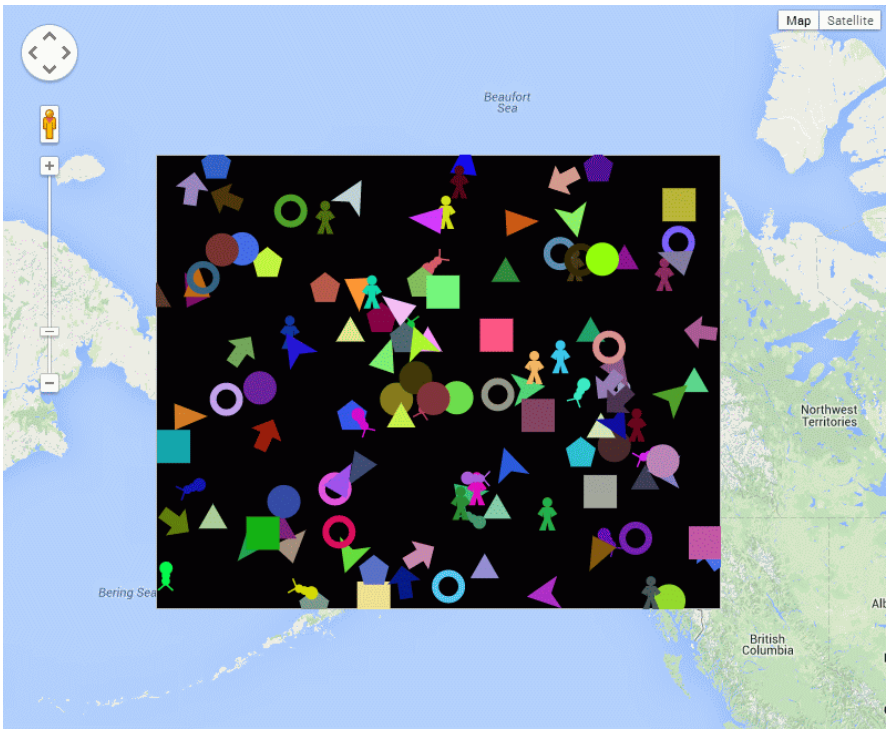


Figure 3.5: AgentScript test model running on top of Google Maps.

Taking this a stage further, figure 3.6 shows a real-time tube model with data taken from the TfL Trackernet API. Archive data from any point in time can be fed into this model to play back previously saved scenarios.

Figure 3.6: Live tube positions, animated using AgentScript, valid for 09:00am on 5 February 2014.

In many situations, simply plotting vehicle positions is not an effective way of presenting information. Real-time data, together with machine learning, can be used to create a knowledge directed visualisation based on deviation from the expected situation. In the case of transport networks, expected wait times can be calculated and deviations from the mean can be highlighted on the map as potential problems. In flight safety maps for aviation, 'colour states' are used to indicate extreme weather which is a hazard to air traffic [Wor15]. Airfields are coloured according to a scale from blue (good weather) though to red (hazardous) based on a combination of visibility, cloud base and wind speed. This is similar to the idea of feature detection in data where learning algorithms are used to identify features of interest. In this example, the features are already 'detected' and based on cloud heights, weather type and visibility which is identified as a hazard to air traffic.

3.10 Spatial Indexing

In the preface of his book, "*Foundations of Multidimensional and Metric Data Structures*" [Sam06], Samet describes the representation of multi-dimensional data as follows:

> "The key idea in choosing an appropriate representation is to facilitate search." (Samet [Sam06, pp. xvii])

Research fields as diverse as robotics, computer aided design, finite-element analysis, solid modelling and GIS are mentioned, as all require an "access structure", or "index" into stored data. Samet makes the point about distinguishing between an "index" and a "data structure", as the term, "index", reinforces the requirement for searching. The following review of the literature relates this to the areas of multidimensional indexing relevant to GIS.

Samet begins his review of indexing structures by posing five questions that determine the properties required of the algorithm:

> "1. What is the type of the data (e.g. is it discrete, is it continuous, is its domain finite)?
> 2. What operations are to be performed on the data?
> 3. Should we organize the data or the embedding space from which the data is drawn?

4. Is the database static or dynamic (i.e. can the number of data points grow and shrink at will)?

5. Can we assume that the volume of data is sufficiently small so that it can all fit in main memory, or should we make provisions for accessing disk-resident data?"

(Samet 2006, [Sam06, pp1])

The first question is about whether the data is bounded or unbounded. In GIS, it would generally be bounded and continuous with finite domain, as the Earth is only a certain size in any coordinate system. In a 3D world application, the world might be too big to represent in one big data structure, so segmentation is used to load hierarchical level of detail chunks into the scene graph as sections of the world become visible. This is addressed by question five. While the world box never actually changes size, the scene graph bounds change as data is loaded and destroyed. The key point here is that the indexing algorithm needs to be able to cope with the change of bounds. Contrast this with the 2D quadtree segmentation algorithm [Kli] [Sam06, sec1.4], used for indexing tiled maps on the web. The method of indexing tiles is to start with the whole world as tile zero, then split equally into 4 tiles for the next level of zoom. These 4 are then split into 4 each, giving 16 and so on recursively (see section 4.2, "Tiled Maps" for a more detailed description). Obviously, if a *"bigger world"* tile suddenly appeared that contained the existing level zero tile within it, then the whole index would need to be rebuilt. Here is where the "operations on the data" mentioned in question two need to be taken into account. Typically, this will be "Add" and "Remove", with different types of index optimised for build once and read many times operations, while other types of index are optimised for record insertion at the expense of read. R-tree and S-tree indexes typify this distinction [Knu00b, ch6]. This covers point four about a static or dynamic database, but the remaining third question, about organising the data or embedding space is more involved. Imagine the data contains points in a finite XY space that are fitted to an 8x8 structure of grid squares (i.e. points on a chessboard). There are two ways of indexing this: either every point is tagged with its square number, or every square stores a list of contained points. Either methodology is valid depending on the statistics of the data being stored and the operations required.

On a related note is the issue of indexing in 2D or 3D data. While Samet discusses multidimensional data in general, in GIS this is usually limited to 2D or 3D physical

space, unless indexing in attribute data, or temporally. A spatial index can be implemented as a 2 or 3 dimensional form of the regular 1 dimensional database index, which is usually a form of R-tree or S-tree representation [36] and so can only index numeric values along a 1 dimensional line. One method of spatial indexing is to use a space filling curve like a Hilbert transform to collapse the 2 dimensional space into 1 dimension and use a quadtree algorithm to grid the data. Methods based on Morton ordering and Peano-Hilbert ordering are compared in [Sam06, pp275]. This type of gridding applied to the data can also be applied at different levels of hierarchy, which is the technique used by Microsoft in their SQL Server spatial indexing.[37] These techniques also have value in the labelling of tiles in recursive quadtree representations for web-mapping, for example in Google Maps.

In general, the type of data being indexed in two dimensions[38] tends to be either points, rectangles, or occasionally spans. Rectangles, or envelopes in GIS terminology, are used as bounding boxes around more complex geometric shapes, generally being axis aligned around the object they bound. This allows for efficient searching of which objects *might* cover an area of space. The concept of a span is sometimes used to bound a set of points along a single axis. Again, these are usually axis aligned spans, for example describing an object that exists between $x = 1$ and $x = 2$ on the x axis.

One feature of spatial indexes which is worth mentioning is that they are often just a heuristic for returning a candidate set of data. Selecting all features which fall within an envelope[39] usually results in some features outside the envelope being returned. The post-conditions for the select function state that the features returned are guaranteed to be at least everything within the envelope, so a second step is required to filter out any extra features returned.

Finally, some implementations of specific types of spatial index need to be defined. In the Geotools library outlined in section 3.5, “Geospatial Libraries”, there are implementations of the quadtree, kd-tree, S-tree and R-tree indexes. While quadtree, S-tree and R-tree have already been defined, kd-tree is relevant to searching for the ‘nearest neighbours’ of a feature, in order of closeness. According to Bentley [Ben75], the

[36] For a mathematical analysis of tree structures see [Knu00b, ch6] and [Sam06].

[37] For details on Microsoft SQL Server’s spatial indexing system see Technet: *http://technet.microsoft.com/en-us/library/bb964712(v=sql.105).aspx*

[38] While the examples here are for two dimensions, this list can be expanded to three dimensions, or more, by suitable modification. For example, rectangles bounding regions become cuboids, then hyper-cuboids to enclose arbitrary geometry. A span remains as a span along a single dimension and a point is always a point in any number of dimensions.

[39] Envelope is the GeoAPI standard terminology for defining a rectangular box.

point kd-tree index is a binary tree representation designed to make a single attribute comparison at each level of the tree. In order to partition 2D space, coordinate comparisons in the x axis are made at even numbered depths with y axis comparisons at odd numbered depths, thus cycling between x and y axis partitioning. This allows searching within bounds, while also enabling an efficient search for nearest neighbours to a starting point.

Chapter 4

Designing Systems: Algorithms and Work Flows

When Google Maps was released in 2005, the traditional way of visualising geospatial data was to download a dataset of interest, join the data with a boundary file where required, and then explore the data on the desktop using GIS software. The advent of web-based mapping started a revolution in the 'publishing' of geospatial visualisations where desktop GIS systems had been dominant before. Continuing this trend, Internet data stores now hold vast quantities of open geospatial data and APIs exist to query this data and other real-time information. This chapter explores how the new architectural elements of web-based maps and Internet data stores are changing the geospatial data landscape.

Before designing any geospatial visualisation system, it is essential that any legal restrictions on the dissemination of the data are understood and followed. Unfortunately, this can often have a notable effect on the design of the system. Taking the situation with the UK 2001 Census as an example, the Census data itself was open data, but the boundary files were protected by Ordnance Survey's copyright and could not be publicly disseminated in their vector form. This led to tiled maps being the preferred medium, as the use of image tiles circumvented the restriction on releasing vector data. The inability to push vector data to the client makes 3D visualisation impossible[1] and results in a thin client solution. With the release of the 2011 Census data, the OS boundary files are now open under the OS Open Data initiative, so this situation has changed.

[1] Only by having the raw vector data on the client is it possible to render 3D visualisations of the data locally. However, a system could be built where the server rendered the scene and passed it back to the client as an image, although every display frame would have to be passed over the network in this manner, making it technologically difficult to implement. Despite extensive research, no system constructed with this architecture has been discovered.

The other defining property of the data being visualised is the quantity. Small amounts of data can be visualised on the client, while the large datasets take time to transfer over a network connection and are likely to be bigger than the client machine can handle. The trade-off here is between small, interactive data visualisation, compared to large-scale visualisations of potentially more interesting data.

Geospatial visualisation in general can be thought of as falling into one of three distinct classifications:

1. Static Maps - pre-rendered, data never changes.
2. Dynamic Maps - rendered on demand, data can change, exploratory.
3. Real-time, or Programmable Maps - data constantly changing, exploration, complex modelling.

In the following sections, each of these architectural decisions is examined in more detail. It should be noted that these sections follow a chronological progression from the first simple web-based maps, through to more advanced solutions as the technology matured, finishing with possible future directions. The aim of this chapter is the requirements gathering phase, where the relative strengths and weaknesses of previous and existing approaches are examined with a view to building a next generation mapping technology.

4.1 "A Place to Put Maps"

When Google Maps was launched in 2005, it made maps and satellite imagery available to the general public for the first time. This paved the way for my 'GMapCreator' tool, recognising the need for a program which could take data from an ESRI shapefile [ESR98] and automatically create a working Google Maps website with it. This is designed for 'intangible' data as opposed to cartography, the distinction being that intangible data does not exist as physical objects in the real world, so population density is intangible, while features like roads and railways are tangible.

By simplifying the process of creating a map, the idea of crowd-sourcing geospatial data was conceived. The next step was the release of CASA's MapTube website at the "Digital Geography in a Web 2.0 World" conference at London's Barbican centre. Figure 4.1 shows the advertising poster:

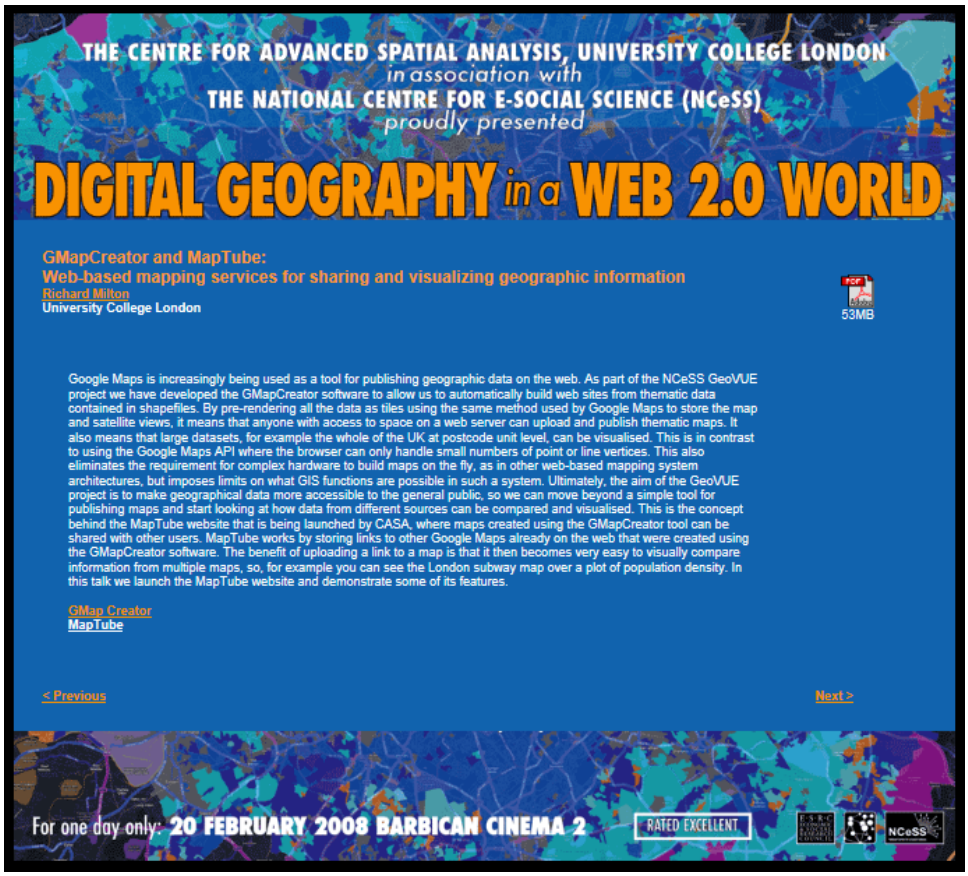


Figure 4.1: The MapTube launch, at the "Digital Geography in a Web 2.0 World" conference at the Barbican centre in 2008: *http://www.maptube.org*.

With the tag line of, “A Place to Put Maps”, the idea behind MapTube is to collect geospatial information, in much the same way as the Environmental Systems Research Institute’s (ESRI) ArcGIS Online product does, although this was released in 2012.[2] The screen-shot in figure 4.2 shows MapTube’s homepage.

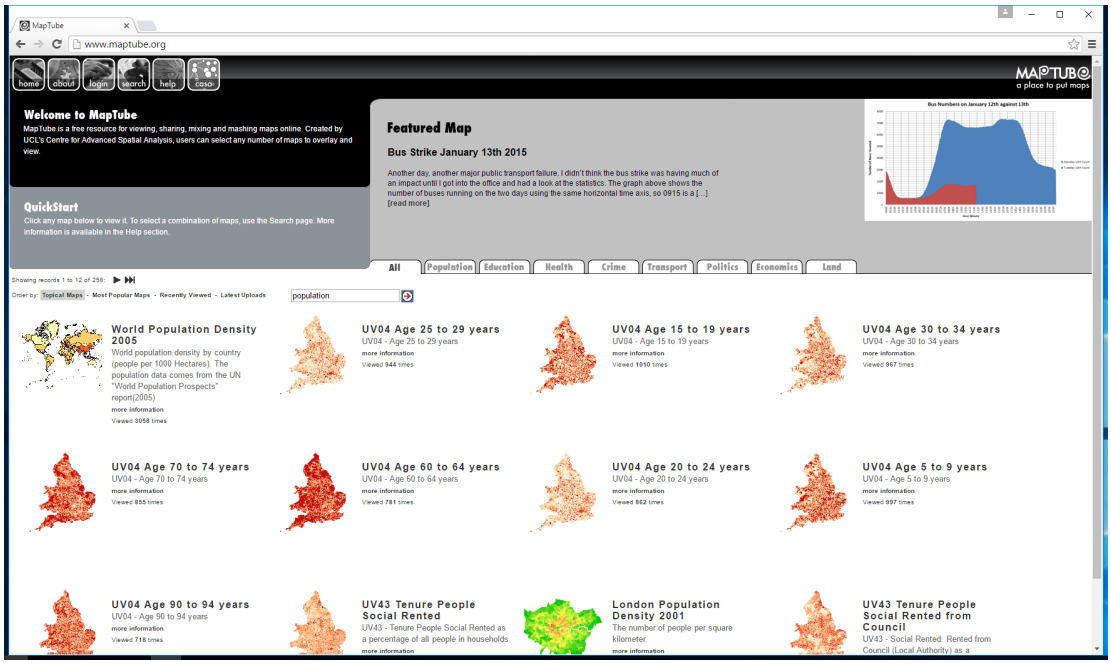


Figure 4.2: The MapTube website’s home page, *http://www.maptube.org*.

Originally, MapTube was designed to share maps which were built with the GMapCreator software. As the graph of the number of downloads in figure 4.3 shows, while there have been 20,928 downloads in total, its popularity has gradually decreased as the technology has changed.

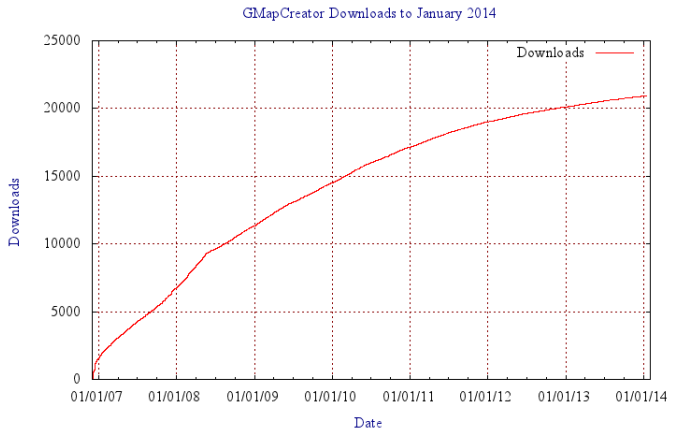


Figure 4.3: GMapCreator downloads from 2007 to 2014.

[2]For ArcGIS online see: *http://www.esri.com/software/arcgis/arcgisonline*.

The theory behind how the GMapCreator makes its static tiled maps is detailed in section 4.2, but it relies on the fact that maps of intangible data tend to be either high resolution and geographically limited, or lower resolution and covering a wider area. In April 2008, this technology was used to run a crowd-sourced survey, in conjunction with Radio 4, on the effects of the 'Credit Crunch'. Here, an online form asked a question of the general public and populated a database with responses which were then used with the GMapCreator and MapTube to draw a new map every 30 minutes. Figures 4.4 and 4.5 show the input form and resulting map.

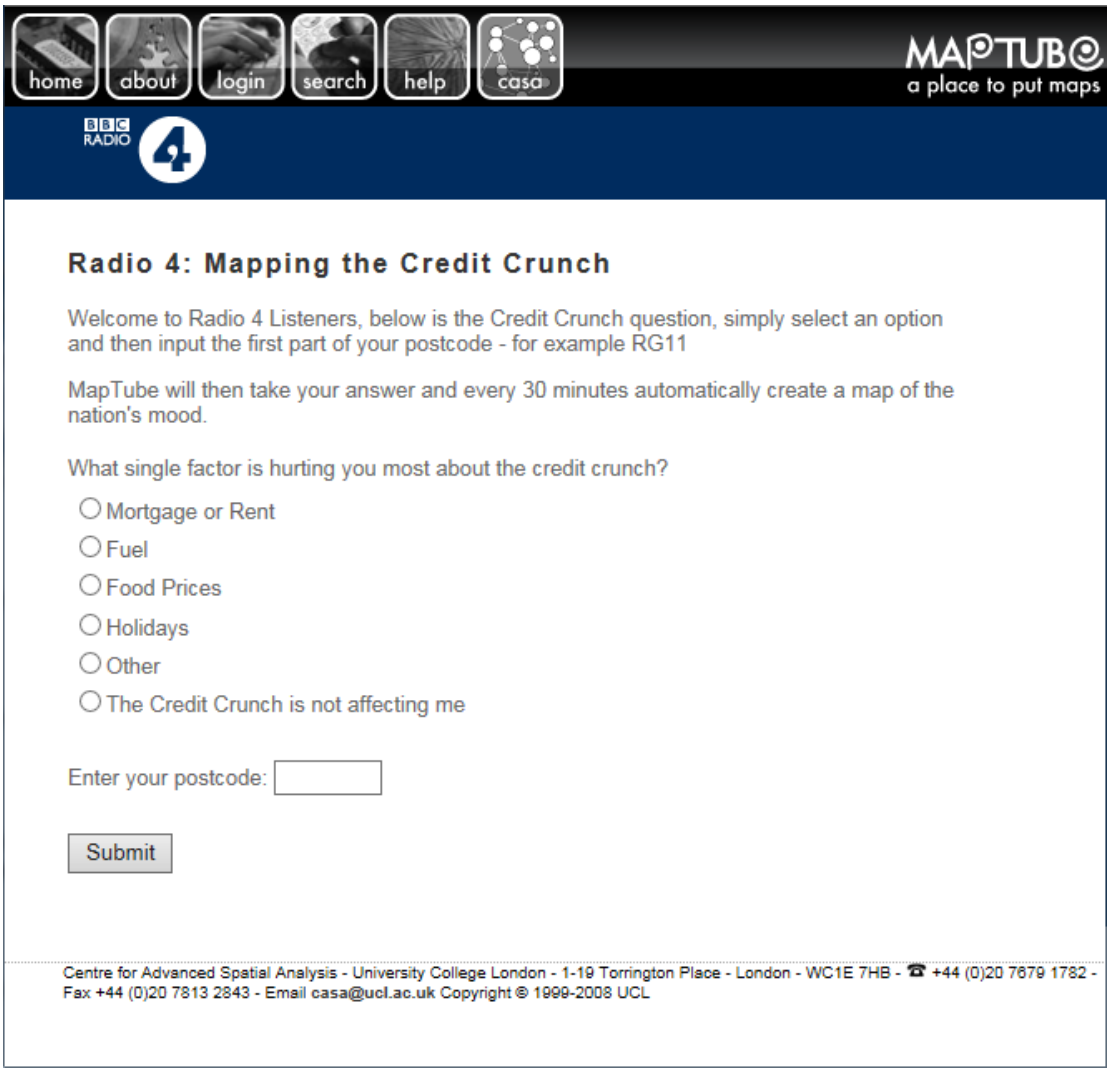


Figure 4.4: Radio 4 Credit Crunch Input Form

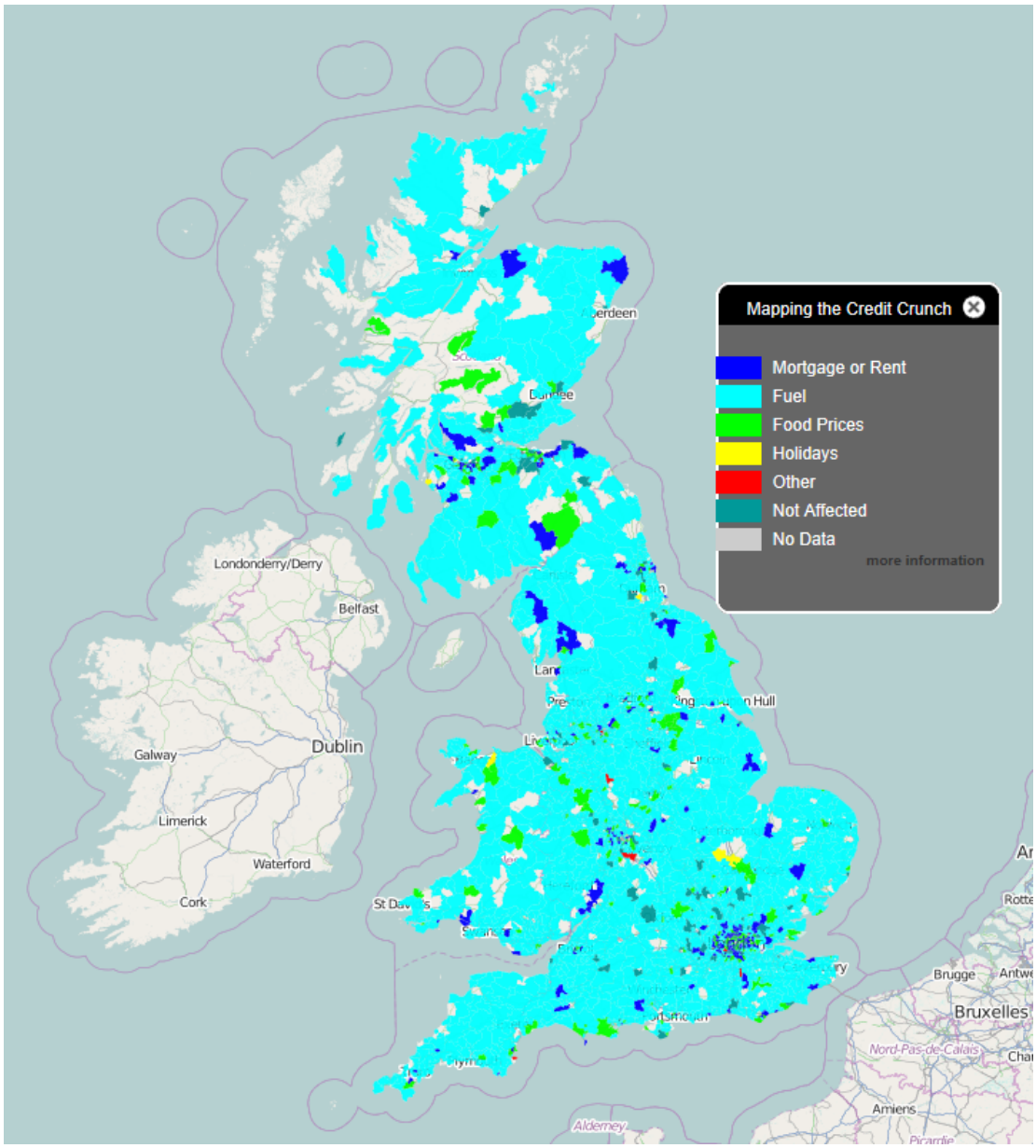


Figure 4.5: Radio 4 Credit Crunch Map

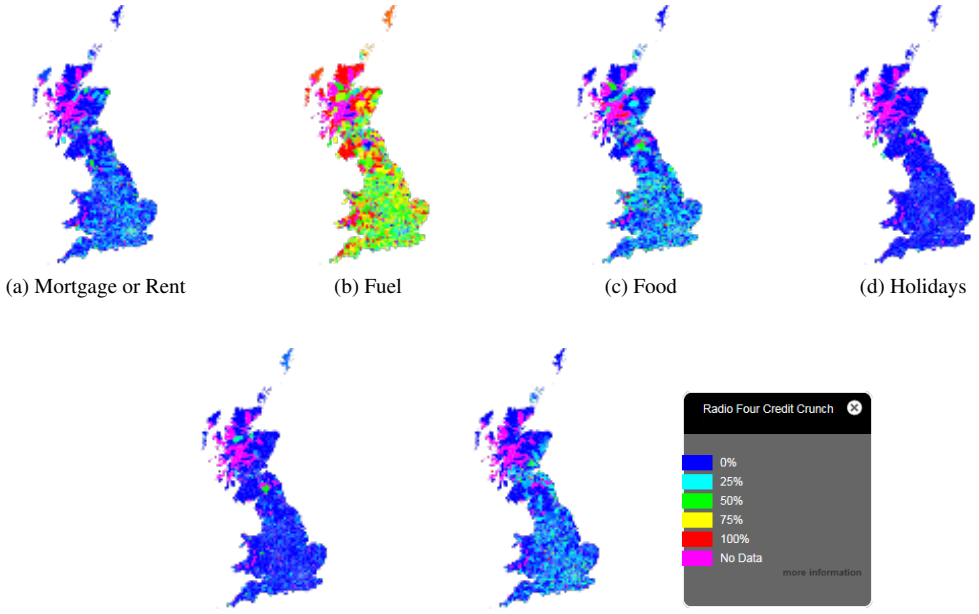
The survey received 23,475 responses because of the publicity from Radio 4, so this was followed by a second survey in October of the same year, receiving 20,072 responses.[3] It was also used by BBC Look East, BBC Look North and BBC South for a variety of surveys on anti-social behaviour, jobs and the recession. The only other major survey using this system was the 'Greater Manchester Congestion Charge' survey which had 15,800 responses, asking what people would do if the proposed congestion charging policy was implemented in Manchester city centre. Out of 'Drive and pay the charge', 'Drive at different times', 'Use public transport/motorbike/bicycle', 'Work or

[3]For the Radio 4 iPM programme page on the MapTube Credit Crunch survey see: *http://bbc.co.uk/blogs/ipm/2008/04/mapping_the_credit_crunch.shtml*.

shop elsewhere' or 'Not affected', not surprisingly, 45.96% of people opted for 'Work or shop elsewhere'. Figure 4.6 shows the BBC's publicity on their own website, which was instrumental in encouraging 15,800 people to respond to the survey.

Figure 4.6: The front page of the BBC News site: *http://news.bbc.co.uk* on Friday 16th January 2009. The map is the second survey for Radio 4, which is about to be knocked off of the top spot by the news of a plane landing in the Hudson river in the United States.

While this is a rudimentary form of web Geographic Information System (GIS), the maps are not updated with people's responses immediately and the ability to query the data on the map is limited. Density maps of each individual response are one further requirement (figure 4.7), but only one map could be rendered in the 30 minute time-frame with the processing power available. When additional features like turning outlines on or off and different colours scales are also added, then the number possibilities makes it impossible to pre-render all the possible maps. This is the difference between a map that is published as a final visualisation and a map that is published as data which the user is expected to explore.



(a) Mortgage or Rent (b) Fuel (c) Food (d) Holidays

(e) Other (f) Not Affected (g) Colour Scale

Figure 4.7: Radio 4 Credit Crunch, percentage of responses for each question per postcode sector. Percentage is calculated based on single question response count divided by total responses.

Section 4.3 covers the structure of the original survey mapping system in more detail, along with the improvements to build the maps immediately. In [Cro+09] the 'near real-time' survey data from the original 'Credit Crunch' survey is analysed and compared with two other social demographic datasets to determine the probable social status of the respondents. This outlines the technical aspects of how the system works, including the system diagram in figure 4.8.

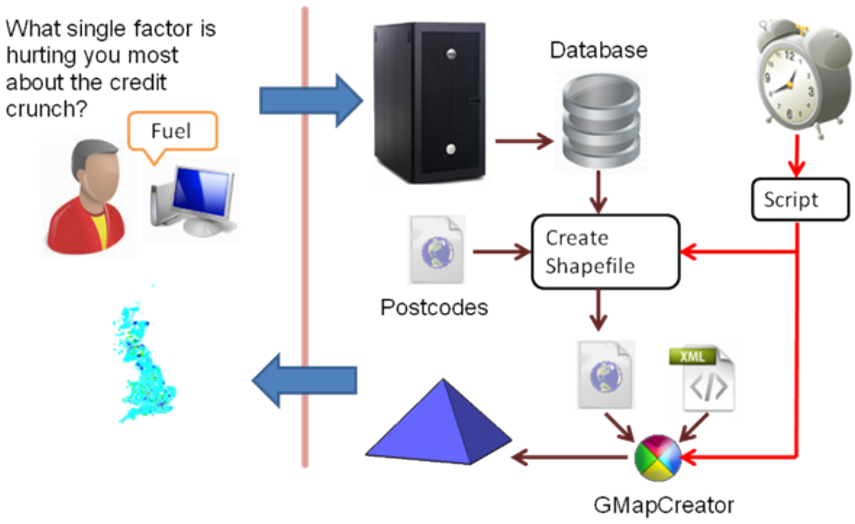


Figure 4.8: Basic system diagram of the near real-time crowd-sourcing system. The blue pyramid represents the tile pyramid of 37,651 map tiles that the GMapCreator builds every 30 minutes.

In "Calibration of a Spatial Simulation Model with Volunteered Geographic Information" [Bir+11] and "Elements of a computational infrastructure for social simulation" [Bir+10], this system is adapted to map the outputs from a population reconstruction model which incorporates the results from the 'Greater Manchester Congestion Charge' survey. The system diagram is shown in figure 4.9, which shows how the automatic mapping capability of MapTube has been adapted to the visualisation of model outputs from a transport scenario. This makes the link between online modelling and online mapping.

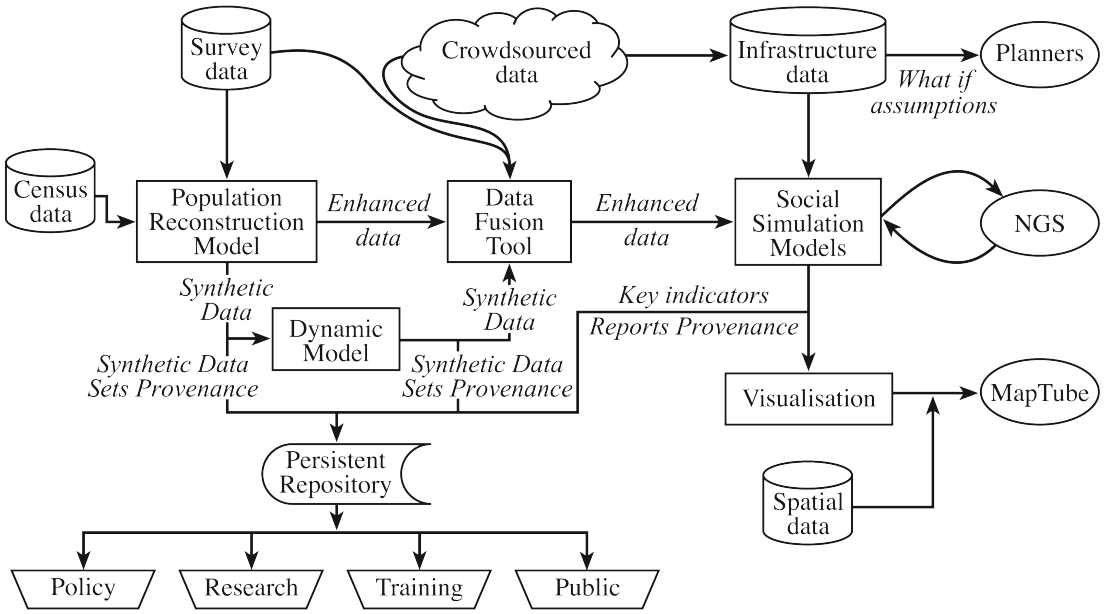


Figure 4.9: The traffic scenario using MapTube to map the outputs of a population model. My contribution to this diagram are the 'MapTube', 'viaualisation' and 'spatial data' boxes and how this links to the social simulation model. Reproduced from figure 1 in [Bir+11].

The 'MapTubeD' system is the next developmental step, which achieves scalability to millions of hits per hour and the immediate mapping of responses to online surveys, which is a requirement for the 'SurveyMapper' website created by Steven Gray in 2010 [GMH15]. The 'MapTubeD' tile rendering layer used by MapTube forms the back-end of the map rendering framework used by SurveyMapper to draw its maps, so providing a re-usable mapping component. The 'D' in the name is a reference to 'Dynamic' mapping, where a set of tile servers can load balance and render data directly from the Internet. This is also covered in detail in Chapter 5, but the technique also gives MapTube the ability to detect the spatial context of data contained in comma separated variable (CSV) files and render the maps automatically.

The sequence of operations required to create a map on MapTube is shown in the

following images. In figure 4.10, a datafile containing the change in population between the 2001 and 2011 Census has been created from the raw data which is available from '*data.gov.uk*'. MapTube takes this data as a text file and data mines it for a column which is recognisable as a geographic area key. This file uses data at 'Lower Super Output Area' level (LSOA), which is one of the standard Census geographies defined by the ONS: *https://www.ons.gov.uk/methodology/geography/ukgeographies*. The data column to map is detected by using statistics to find the first numeric column that is not an arithmetic progression. Numeric columns are allowed to contain a single alpha numeric value representing a "no data value", but this must be consistent.

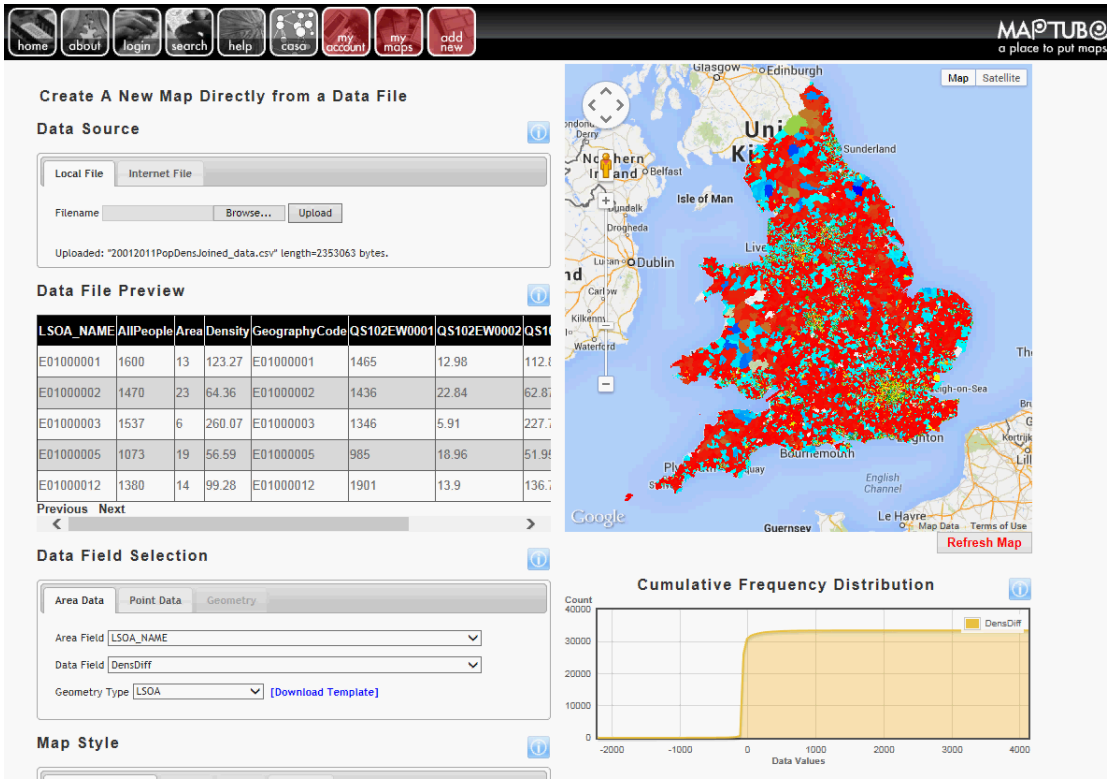


Figure 4.10: MapTube creating a map automatically from data contained in a CSV file.

In the next image, figure 4.11, the screen is scrolled down to show the 'Area Field', 'Data Field' and 'Geometry Type', along with the default colour scale and cumulative density plot of the data. The idea behind the design was to allow the user some control and flexibility in choosing what data to map and how it is displayed. The cumulative density plot shows the type of data being viewed, which also dictates the choice of colour scale. In this example, the data is a continuous positive value containing counts, but could be data distributed about a mid-point, or a small set of discrete class values. The next step is to choose a more appropriate set of colours to show the data.

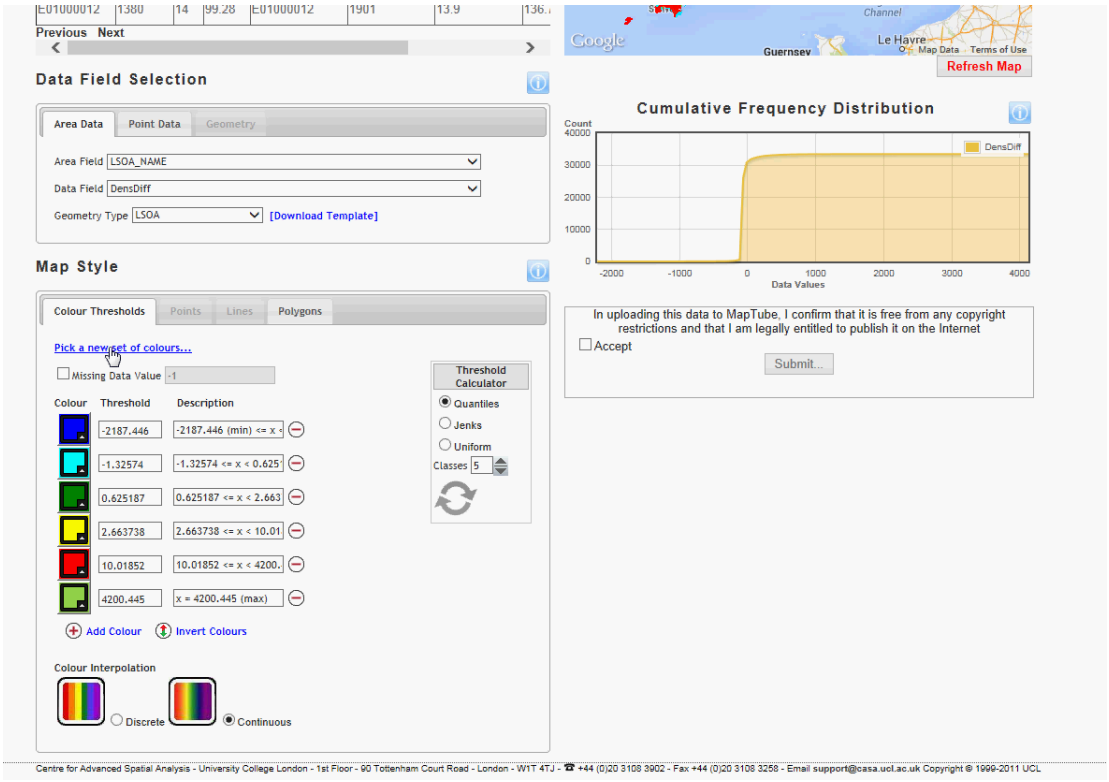


Figure 4.11: Choosing the data field to map, the colour scale and breaks type (quantiles, Jenks or uniform). The discrete or continuous colour interpolation option controls whether colours are interpolated based on data value.

The sets of colours available in the dialogue shown in figure 4.12 are divided into: 'Sequential', 'Diverging' or 'Qualitative'. For a map of continuously varying sensor values, for example carbon monoxide levels, the sequential scale would be chosen. For the population data, there is a natural divergence in the data about zero, where population has increased or population has decreased. A red and green scale with white in the centre makes sense for this type of data. Qualitative scales are used when data falls into discrete classes as was the case with the survey mapping examples earlier. The colours come from Cynthia Brewer's set, which can be found online[4]. These colours have become a de-facto standard in GIS as they were designed to reduce the possibility of errors due to misinterpretation of the colour both on computer screens and with printed maps under varying lighting conditions. In this example, the transition between red and green has been chosen.

[4]ColorBrewer2: *http://colorbrewer2.org*.

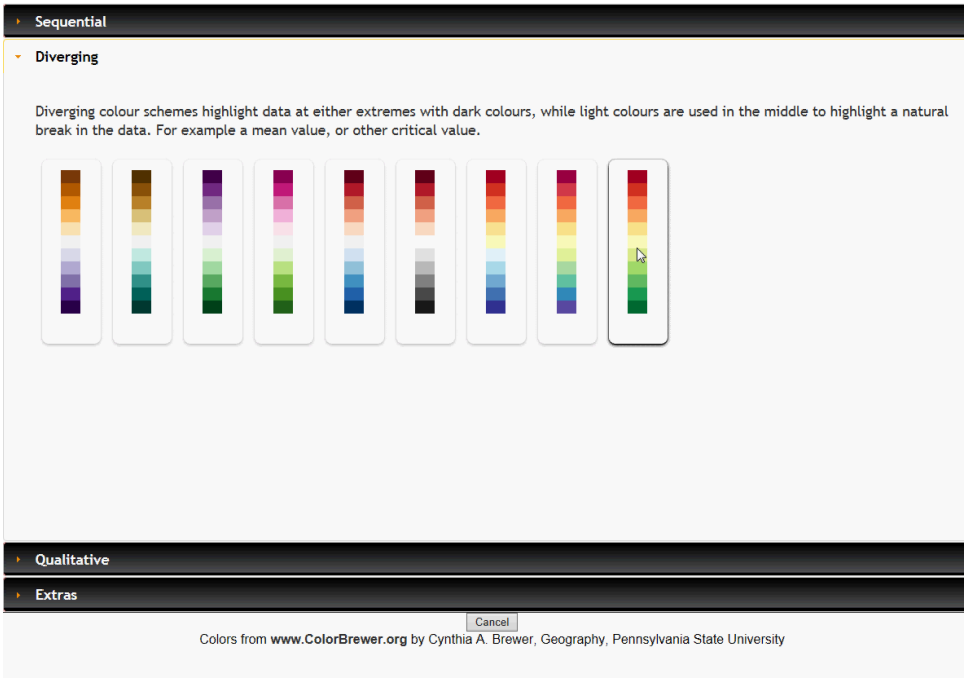


Figure 4.12: Choosing a diverging red to green colour scale for the population data.

The map in figure 4.13 now shows the result of the change in colour scale where red shows population decrease and green shows population increase.

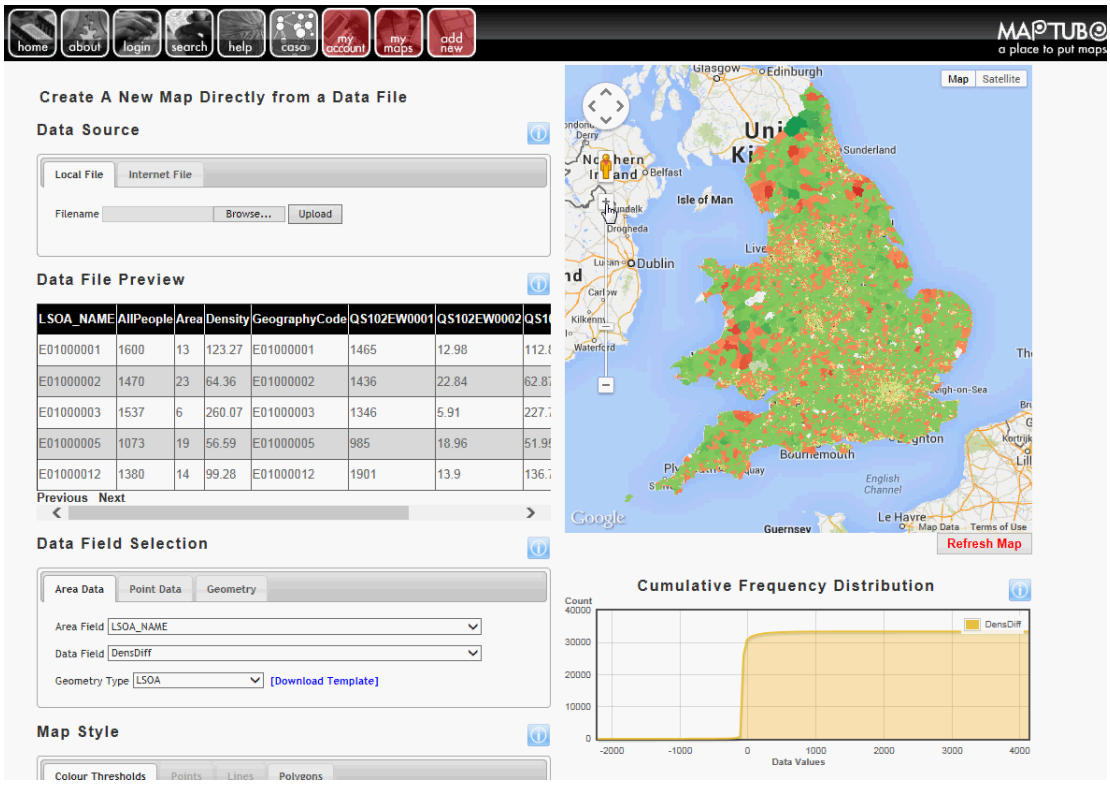


Figure 4.13: The map now shows the data using the red to green colour scale.

Scrolling the web page down to the bottom shows the changes which have been made to the colour scale colours (figure 4.14). These can all be adjusted by the user simply by clicking on the button and picking a new colour if required. The breaks types can also be changed at this stage, either by entering new values into the threshold boxes, or by switching from quantiles to Jenks natural breaks, or a uniform scale depending on the type of data.

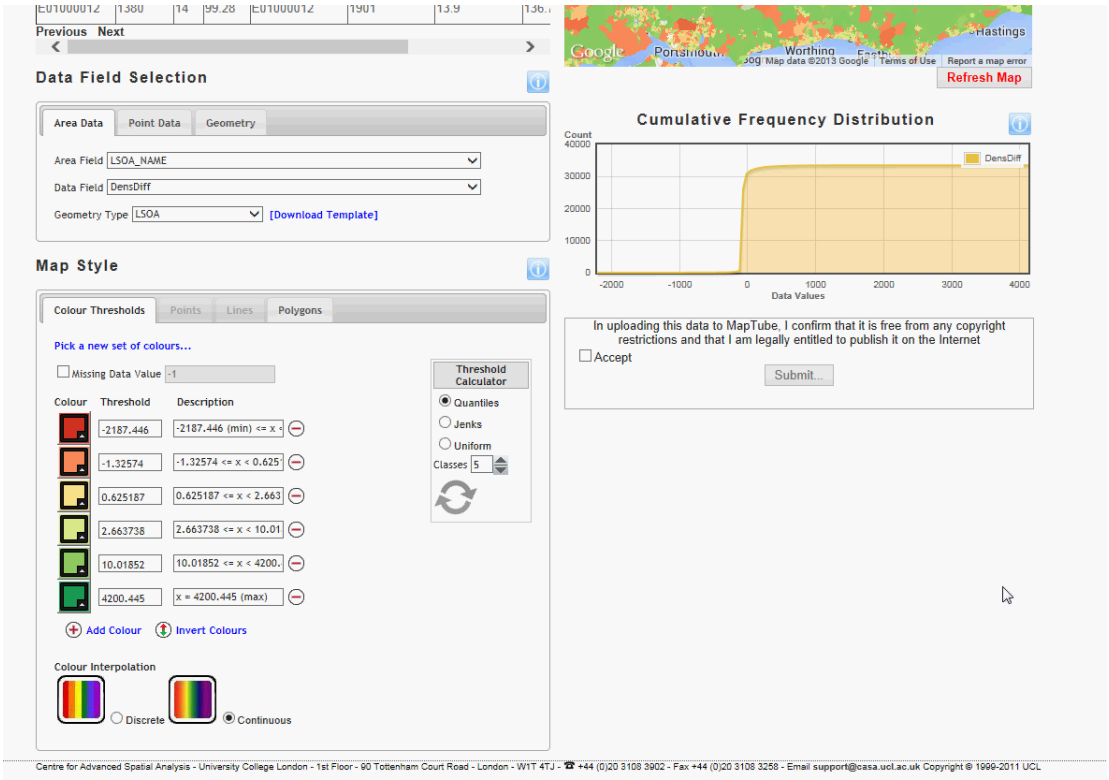


Figure 4.14: Colour scale and breaks ranges and type for the population change map.

The final image in figure 4.15 shows the resulting map published on MapTube and accessible by anyone visiting the site. The MapTube homepage also has a 'Latest Uploads' link which shows all the maps which have been created recently. In addition to this, there is also a topicality index which is built from the BBC's RSS new feed and is designed to promote any maps currently in the news to the front page of the site. Although searching by keyword is provided as a method for finding information, the question of how to search through a maps data store containing several thousand maps remains a problem. The map in this example is also 'clickable' by default. With this dataset, clicking on one of the areas triggers a popup on the map which contains all the Census fields for that particular area.

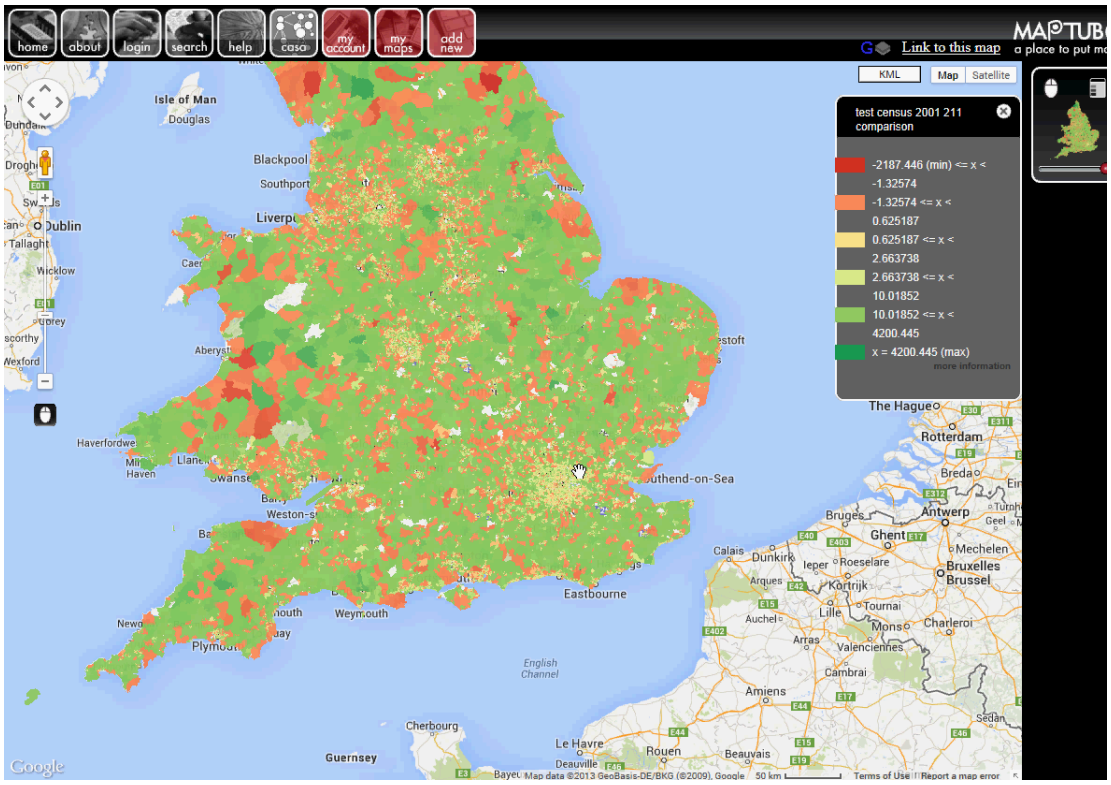


Figure 4.15: The final population density difference map published on the MapTube website.

4.1.1 Topicality Index

Algorithm 1 describes how MapTube calculates a "topicality index" for each of its maps. This is used to promote data that is currently in the news, but it could also be used in reverse to identify where information is lacking. The idea that you can detect what you do not know and spot gaps in a body of information is an interesting one. Taking all the London maps as an example, using the spatial context of the data, then any news stories about London not matching any maps would hint at missing information.

The first stage of the topicality index algorithm begins by building a "bag of words" list for all the words contained in the text of the title, description and xml content fields for all of today's posts for every one of the RSS feeds on the feed list. As the content is html, all of the tags must be stripped from the data to turn it into plain text, before the text is split into a list of words. During this process, any words of less than 3 characters are dropped, along with non-alpha characters. The word split is performed on the space, comma, full-stop and newline characters.

Having created a list of words from current news articles, the next stage turns this into a dictionary of unique words and frequency. Here, the Porter Stemming algorithm is used so that only the stem of the word is used as the unique key [JW97] [Por80]

[Por91]. For example, if the text contained the words, "computer", "computers" and "computing", then all three are reduced to the stem of "compu". Compared to vector embedding techniques like Mikolov's "Word2Vec" [Mik+13], stemming is basic, but effective. Also, the implementation developed here pre-dates Mikolov's original paper on word vector embedding by two years. Finally, in the algorithm pseudo-code there is reference to "spoof" keywords. After implementation on the live server, it was discovered that certain combinations of keywords appearing in the media caused unusual maps to appear at the top of the list. This led to the addition of manual weightings applied to words, for example the keyword "test" appearing in the media is weighted down in the following section, otherwise it causes lots of *test* maps to appear on the homepage, where the user probably hasn't finished uploading the data for them yet.

To complete the topicality calculation, the stemmed words from the media are transformed into probabilities by normalising by total word count. Then they are matched up to a set of stemmed words generated from the title, short description, long description and keywords stored for every MapTube map. This now means that there is a word probability for the media words and a count of word frequency for the words describing MapTube maps. The topicality index can then be calculated for every MapTube map by taking each map's word list in turn and scoring it against the media word list.

$$T_m = \sum_{i<|W_m|)} \frac{W_{mi} W_{nj}}{\sum_{j<|W_n|} W_{nj}} \qquad \text{where } W_{mi} \text{ and } W_{nj} \text{ are the same word} \tag{4.1}$$

T_m	topicality index for map m
W_m	list of words describing map m e.g. ["this", "is", "a", "map"]
W_{mi}	word i in the list e.g. $i = 3 \implies$ `map` in the example above
W_n	list of words in all news articles
W_{nj}	word j from the list of words in news articles

Although equation 4.1 gives the basic equation, this is calculated for each map in turn using the words from the "title", "keyword" and "description" fields separately, with the results added together to give the final topicality index as follows:

$$T_m = T_m^{title} + T_m^{keywords} + \frac{T_m^{description}}{2} \tag{4.2}$$

The reason for the 0.5 weight on the description is that it contains plain text describing the map, with many more words than the title and keywords. The additional weighting was an empirical fix to make this work effectively in practice. The title and keywords have greater weight in describing what the map is about, while the plain text description is less focussed.

Finally, the topicality index, T_m, for every map is obtained, but in addition to this, algorithm 1 also shows the words and counts for each map that went into this calculation being returned in $TopicalWords$. Not only does this provide an explanation for the topicality index values, but it can also be used as a cross-check against the words from the news media, W_n, which were *not* matched against any maps. As stated at the start of this section, any media words not matched against any maps suggest a need to locate and add new data. For example, if the word "Spanish" appears and there are no Spanish maps on MapTube. Given the automatic nature of the data mapping system developed for MapTube one idea was for the system to be extended to find its own data on the Internet.

On a related point, the data from the topicality keyword matching can also be reused in another way. Rather than matching the words describing maps to news items, the maps can be related to each other by number of shared words and word probability. This is future work, but a form of graph search could be constructed to find similar maps based on concept, rather like Internet recommendation systems based on association rules.

Algorithm 1 Topicality Index

Require: A list of RSS sites: e.g. BBC, Guardian, CASA Blogs
RSS_Keywords $< string, int >\leftarrow$ *empty hash*
for all *RSS_Site* **in** *RSS_Site_List* **do**
  for *post* **in** *RSS_Site* **do**
    for *xml* = $(title, keywords, description)$ **in** *post* **do**
      text $\leftarrow$ html stripped from *xml*
      words $\leftarrow split\_words($*text*$)$
      for *word* in *words* **do**
        if not$(is\_stopword($*word*$))$ **and** *len(word)* > 2 **then**
          $RSS\_Keywords[word] \leftarrow RSS\_Keywords[word] + 1$
        end if
      end for
    end for
  end for
end for
Post-Cond: RSS_Keywords contains a list of keywords from the RSS feed, along with a count of the number of times each word appears
StemWordCounts $< stem\_word, int >\leftarrow$ *empty hash*
for all *word*, *count* **in** *RSS_Keywords* **do**
  Create *StemWord* from *word* using Porter Algorithm
  Note: two words can result in the same stem word, so StemWordCounts < RSSKeywordCounts
  if *StemWordCounts* **contains** *StemWord* **then**
    StemWordCounts[StemWord] $\leftarrow$ *StemWordCounts[StemWord]+count*
  else
    StemWordCounts[StemWord] $\leftarrow$ *count*
  end if
end for
TotalCount $\leftarrow$ sum of all word counts **in** *StemWordCounts*
(*Normalisation step*)
$StemRSSKeywords \leftarrow StemWordCounts/TotalCount$
Add spoof keywords and counts to *StemRSSKeywords*
$MatchedKeywords \leftarrow$ hashtable matching mapid, title, keywords, desc to $StemRSSKeywords$
(*Topicality calculation*)
$Topicality < int, float >\leftarrow$ every map id from MapTube (int) with a zero value (float)
$TopicalWords < int,$ *List of StemWords* $>\leftarrow$ every map id from MapTube (int) with an empty word list
for all mapid in Maps **do**
  $T\_title \leftarrow StemRSSKeywords * MatchedKeywords(mapid, title).count \ \forall$ *word matches*
  $T\_keywords \leftarrow StemRSSKeywords * MatchedKeywords(mapid, keywords).count \ \forall$ *word matches*
  $T\_desc \leftarrow StemRSSKeywords * MatchedKeywords(mapid, desc).count \ \forall$ *word matches*
  $Topicality(mapid) \leftarrow T\_title + T\_keywords + 0.5T\_desc$
  $TopicalWords(mapid) \leftarrow$ all matching words
end for
return $Topicality, TopicalWords$

4.1.2 Internet Maps

Shortly after this was released on the MapTube site, Google launched their own 'Fusion Tables' application, which is very similar in concept, but differs in one subtle detail. While Fusion Tables needs the data to be uploaded to a Google account, MapTube was designed from the outset to use data directly from the Internet. This feature is exploited later in section 4.4.1, where the concept of 'mining' an Internet data store for maps is introduced, resulting in a worked example using the UK's 2011 Census in the data exploration of Chapter 7. 'Data store mining' is an appropriate name for this technique as, after collecting data on MapTube for 8 years, a valid question to ask is, "How do all the maps relate to one another?". This question is answered using the Census 2011 data in Chapter 7. One final point to note about MapTube is the question of completeness. In order to vary the maps shown on its front page, MapTube mines the text in news stories published on a number of media websites. It uses this information, along with the text descriptions of its maps, to build a topicality ranking. The maps that show on the front page should then be related to current news items in the media, but the technique can also be used in reverse. Anything that ranks highly in the media, but does not have an associated map, suggests possible missing data. Network graphs of how maps are linked in both the data and meta-data domains give structure to the data, a concept that is explored in Chapter 5.

The final piece of infrastructure concerns real-time data. 'Application Programming Interfaces', or 'APIs', describe mechanisms by which user programs can access data or functionality provided by a third party. Through moves towards open Government, many sources of real-time data are now accessible to the general public. London Underground, for instance, makes information about the tube system freely available so that developers of mobile phone Apps will create utilities like 'CityMapper'[5] which helps their customers. Another company, 'TransportAPI',[6] provides unified access to transport data in the UK, bringing together a number of real-time data feeds into a single interface. Before either of these were created, though, I created the 'Adaptive Networks for complex Transport Systems', or ANTS, project to provide real-time data to MapTube and subsequent systems like '*citydashboard.org*'. Along with a 2 dimensional visualisation as a map on MapTube, a 3D, browser based, visualisation was created.

[5]CityMapper is a mobile application which was developed to help people navigate around cities: *http://www.citymapper.com*.

[6]TransportAPI was originally called 'PLACR', which was a spin-off from City University. Their website is: *http://www.transportapi.com*.

Figure 4.16 shows a real-time WebGL view of tube trains and buses in the Chrome web browser.

Figure 4.16: A real-time view of tube and bus positions using the Google Chrome web browser. The red cubes are buses and river services, while rectangles showing live tube train positions are shown in the regular TfL line colours.

The real-time data on tube and bus numbers, along with air quality, was used for the 'iPad Wall' project which was installed in the London Mayor's office in 2012 [GMH13]. The server which collects the real-time data also archives it, which provides the data for Chapter 6. Part of the motivation for writing this thesis is to use the 3GB a day of real-time data that has been collected since July 2012 to understand how the different city systems interact with one another. The incorporation of both static and real-time data is a key element. This forms the bulk of the data exploration in Chapter 8.

After creating real-time visualisations of tube positions and graphs of weather data and air quality, the realisation is that more advanced analytics are required to turn the information into knowledge. Taking the map in figure 4.17 as an example, this shows the mean wait times at stations on the London Underground network.

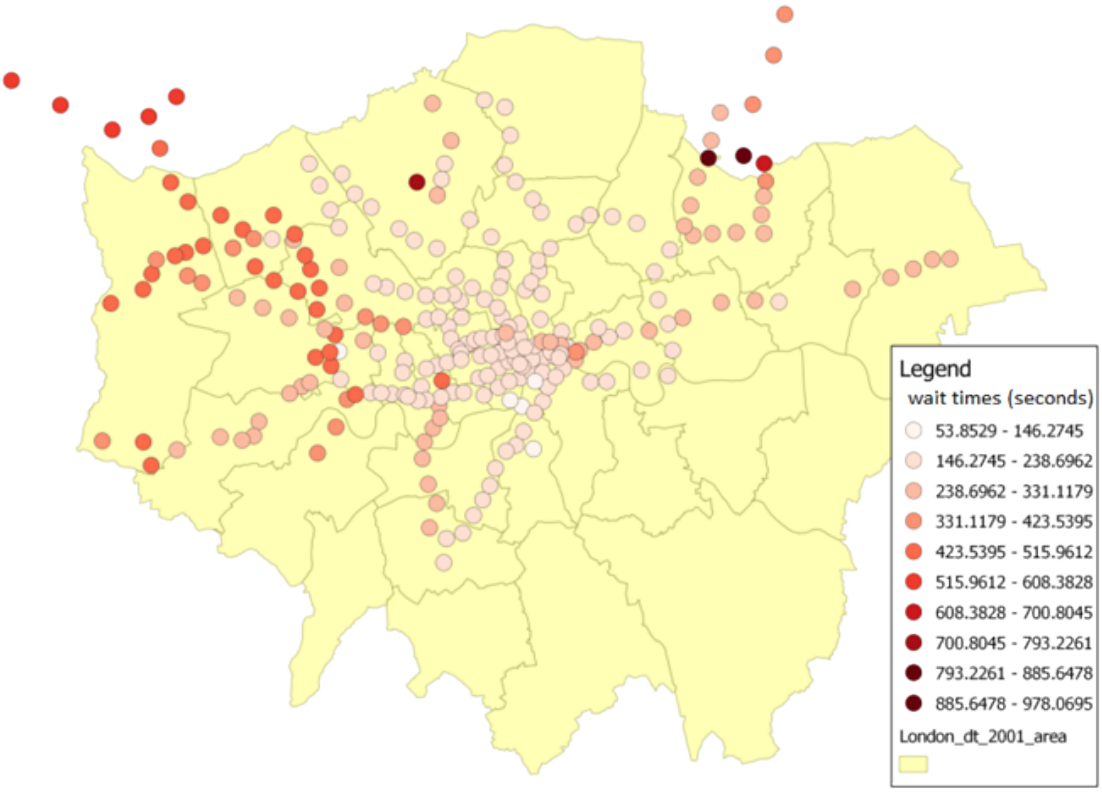


Figure 4.17: Mean wait times for London Underground stations plotted using QGIS.

By detecting where the real-time running data shows a wait time more than one standard deviation above the mean, unusual events can be detected. The approach is one of tightly coupling agent based modelling and data mining with real-time data and visual analytics. The solution presented in Chapter 6 is to learn from the archive data and ground the model using the derived statistics.

To conclude, the aim of this thesis is to further develop web mapping technology into a true web GIS, capable of analysing both static and real-time data. After 9 years developing the MapTube website to collect data and make maps, the goal is now to develop new ways to explore and analyse the data.

4.2 Tiled Maps

The basic premise behind a tiled map is that it starts with a single square image, then divides this into four, and continues dividing recursively using a quad tree segmentation algorithm, so that each successive zoom level contains four times as many images, or tiles, as the previous. The map scale is therefore decreasing with each successive zoom level, so more detail can be seen as the user zooms in.

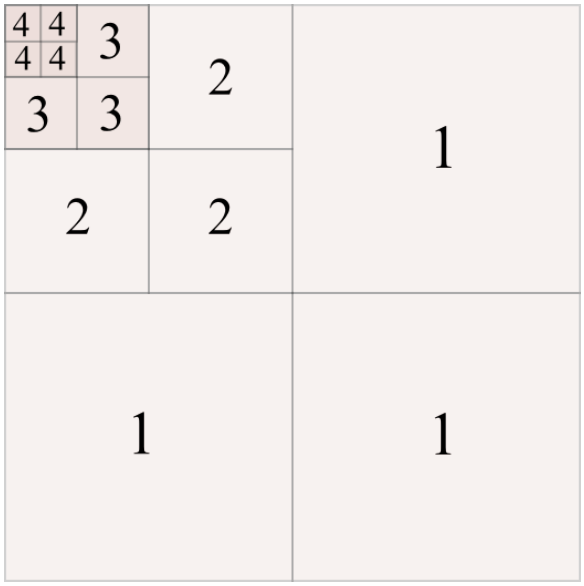


Figure 4.18: Tiled map squares with the top left square recursively divided. The numbers show the level of zoom.

Figure 4.18 shows the recursive subdivision of the top left tile. Further information about tiling and how it was used to create static tile sets for London Census data can be found in [Gib+08] which details the development of the 'London Profiler' website. For a more general view of web mapping, [Bat+10a], covers map 'mash-ups', where data from different sources is overlayed on a map, along with other aspects of web mapping past and future.

Tiled maps used for web mapping which conform to the original Google tile specification, or the retro-fitted OpenGIS "Web Map Tile Service" described in section 3.6, quadruple in number at each successive zoom level. An estimate of the number of tiles required to store the whole world can be made by summing the total number of tiles over all zoom levels and applying a linear correction factor to take into account the letterbox effect of the -90 to +90 latitude and -180 to +180 longitude rectangle. At the top level, the whole world is represented by one tile, which is split using a quad tree algorithm. At the next level this splits into four tiles, followed by each of them splitting

into four to make sixteen. This is repeated for every subsequent level of zoom, yielding a formula for the total number of tiles as follows:

$$N = 4^0 + 4^1 + 4^2 + \ldots + 4^n \qquad \text{where } n \text{ is the final zoom level} \tag{4.3}$$

$$N = \sum_{z=0}^{n} 4^z \tag{4.4}$$

Although equation 4.4 can be calculated for each zoom level individually, it can be reduced further by recognising that it is a simple geometric progression of the form:

$$N = \sum_{k=0}^{n} ar^k = ar^0 + ar^1 + ar^2 + \ldots + ar^n \tag{4.5}$$

$$N = \sum_{k=0}^{n} ar^k = \frac{a(1 - r^{n+1})}{1 - r} \tag{4.6}$$

$$N = \frac{a(1 - r^{n+1})}{1 - r} \tag{4.7}$$

In this case, $a = 1$ and $r = 4$ to give the quad tree progression. Taking $n = 17$ as the maximum zoom level and using equation 4.7, gives an initial estimate of:

$$N = 22,906,492,245 \;\; tiles \tag{4.8}$$

This needs to be divided by 2 because only the tiles within the (-180,-90), (180,90) envelope are stored. Then a correction factor needs to be applied to account for missing tiles in remote locations, for example the middle of the ocean, which are also not stored. As this is an estimate, a linear scaling factor can be applied to all zoom levels uniformly, so the requirement is to calculate the percentage of the Earth that is water. Although more complex methods exist to calculate the area of complex polygons, for example the strip method introduced in [SP98], as this is an estimate, reprojecting the data using an area preserving projection and simply calculating the water to land ratio is sufficient. World vector maps exist which are free to download, for example, 'TM_WORLD_BORDERS-0.3' from Bjørn Sandvik's website, '*thematicmapping.org*'. This particular shapefile happens to contain the United Nations Food and Agriculture Organisation's data on country areas in thousands of hectares, so it is a simple

matter to add up all the areas in the attribute table and obtain a total in KM^2.

The formulas to obtain the surface area of an oblate spheroid are widely published and a quick search of Wolfram Mathworld [Wei14] yields the following:

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{4.9}$$

$$S = 2\pi a^2 + \pi \frac{b^2}{e} \ln\left(\frac{1+e}{1-e}\right) \tag{4.10}$$

Using the WGS84 coefficients:

$$\begin{aligned} a &= 6378.137KM && (semi-major) \\ f &= 1/298.257223563 && (flattening) \\ b &= a(1-f) = 6356.752KM && (semi-minor) \\ S_{oblate} &= 510050577.7KM^2 && (from\ \ equation\ \ 4.10) \end{aligned}$$

Dividing the land area from the shapefile by S_{oblate} results in the following estimate:

$$Land = \frac{129632390}{510050577.7} \times 100\% = 25.4\% \tag{4.11}$$

Working on the basis that 25.4% of the world is land[7], the result in equation 4.8 can be modified as follows:

$$N = \frac{1}{2} \times \frac{25.4}{100} \times 22,906,492,245 = 2,909,124,515 \tag{4.12}$$

As the original question related to storage of all the tiles comprising the world, the next problem is to estimate the size of a tile on disk. Web mapping systems have almost exclusively used 256 pixel square tiles as a trade-off between size and detail, so an upper bound for the size of a tile can be estimated as follows:

$$256px \times 256px \times 32\ \ bits\ \ per\ \ pixel = 256KB \tag{4.13}$$

This figure assumes a worst case scenario and no compression, while by averaging all the tiles actually stored by a web mapping system, an average of 2KB has been observed in practice.

[7] The National Oceanic and Atmospheric Administration (NOAA) states that the percentage of Earth's surface covered by water is 71%, so the estimate of 25.5% is too low, but it is important to show how to estimate the area on the WGS84 spheroid. The reason the estimate is low is because not all the land masses are accounted for in the data. See: *http://www.noaa.gov/ocean.html*.

$$N \times 4KB \;\; per \;\; tile = 11,636,498,060KB = 11,097GB \tag{4.14}$$

This is significantly bigger than a 33GB Open Street Map 'world file' which stores all the data for the planet, and is worth rendering only on demand, especially when it is realised that this estimate is only for a single map. While there might be a limited number of cartographic representations in use, when more general representation of data on maps is considered, the number of tiles multiplies. Taking the 2011 Census as an example, there are 2,558 variables which can be mapped (from the NOMIS r2.2 release), while for the 2001 Census, a related project called the "London Profiler"[8] mapped population and deprivation statistics on Google Maps using pre-created static tiles [Gib+08]. This comprised a subset of individual Census variables, mapped down to zoom level 18 for the limited London area. These scales of magnitude are important to bear in mind when attempting to make maps from all the data contained in entire data stores. In general, with pre-created tiles, it is necessary to limit either the zoom level, or geographic area.

Despite the fact that the Mercator projection has singularities at the poles, which require the map to be truncated above and below certain latitudes, it is still used almost exclusively in zoom-able web-based mapping systems. While the projection allows for a non-uniform aspect ratio, web maps define the aspect ratio to be unity, which, along with the spherical Earth used as the datum, results in the truncation of the maps according to the following formula. The symbol, ϕ, represents latitude, while R is the Earth's radius and y is the Mercator North ordinate.

$$\begin{aligned} AspectRatio &= 1 \Rightarrow y = \pm\frac{width}{2} \\ \frac{y}{R} &= \pi \\ \phi &= \tan^{-1}\left(\sinh\frac{y}{R}\right) \\ &= \tan^{-1}\left(\sinh\pi\right) \\ &= 85.05113^{\circ} \end{aligned}$$

While a map truncated at ± 85 degrees latitude is fine for most applications, where many environmental maps are concerned, not being able to represent data near the poles

[8] London Profiler website: *http://www.londonprofiler.org*. Unfortunately, this is no longer online, so only exists in [Gib+08].

is unacceptable. The key issues with this type of system are to be able to reproject the data into the Mercator projection, then each tile request results in a box test to select the data to be drawn on the tile. This involves calculation of the tile box from the tile coordinates passed in the web request, a spatial index to optimise selection of the data, followed by rendering to the tile. The rendering of the data can be performed by a geospatial library like geotools (Java), SharpMap (C#) or Mapnik (C++), with the lower level geometry handled by a library like the Java Topology Suite (Java), NetTopologySuite (C#) or GEOS (C++). The GEOS library was used by me as part of the 3D pipeline to take a map asset from MapTube and place it in the "Second Life" online system using Autodesk's FBX format as published in [Hud+09a], [Bat+09] and [Bat+10a]. Any system able to handle 2 dimensional path geometries with holes and rasterise them is a suitable candidate, including GPU implementations.

The WMTS standard from the OGC is a response to the proprietary web mapping standards that have emerged as tiled maps have gained popularity. By developing this new standard, the OGC addresses the original WMS standard's inability to scale to large numbers of users. Quoting from the OGC WMTS specification document:

> "The OGC WMTS provides a complementary approach to the OGC Web Map Service (WMS) for tiling maps. WMS focuses on rendering custom maps and is an ideal solution for dynamic data or custom styled maps (combined with the OGC Style Layer Descriptor (SLD) standard). WMTS trades the flexibility of custom map rendering for the scalability possible by serving static data (base maps) where the bounding box and scales have been constrained to discrete tiles. The fixed set of tiles allows for the implementation of a WMTS service using a web server that simply returns existing files. The fixed set of tiles also enables the use of standard network mechanisms for scalability such as distributed cache systems."
>
> (From document 07-057r7_Web_Map_Tile_Service_Standard [Ope10a])

Part of this statement from the OGC's standard could be seen as contentious though, as there is no issue with serving dynamic data using WMTS. The ability to cache map requests and the ability to handle dynamic data are two separate issues. Taking Google's Fusion Tables system as an example, this certainly does work with dynamic data, while real-time mapping systems like CASA's 'SurveyMapper'[9] can build dy-

[9] Survey Mapper website: *http://www.surveymapper.com*.

namic maps on the fly using volunteered geographic information.

Following the Google release, both Microsoft and Yahoo! followed with their own mapping systems and MetaCarta began work on an open source alternative to the commercial Javascript map libraries. Their 'OpenLayers' project performs the same function as the Google Maps API, relying on an available source of map tiles in one of a number of popular formats. This highlights one of the problems with the development of tiled map systems. Where no officially recognised standards existed, third party vendors made up their own. Google initially used Keyhole strings made up of the letters 't' (NW), 'q' (NE), 'r' (SE) and 's' (SW) to define tiles, but quickly standardised on numbering in the X and Y directions with an additional zoom parameter, as 'Z_X_Y'. While they chose to number with the origin in the top left, other mapping systems used a bottom left origin, or specified tiles as strips in directories as 'Z/Y/X' which was favoured by OpenStreetMap.

One of the enabling technologies involved in the propagation of maps on the web is the existence of data. Google's initial business insight was to licence the data from mapping agencies all around the world to build a connected map which was then freely available to view. Both Google and Microsoft now spend billions of dollars on their mapping services, which they finance partly though advertising and premium services.

With the OpenStreetMap community actively mapping the whole world, open vector data to make maps is now freely available. The missing link is currently the satellite data, street view images and 3D maps which do not exist in any open form yet. While looking at 3D representations of maps, for example Google Earth or NASA's WorldWind, rendering on the user's computer requires the raw data, which requires licensing of the data in an appropriate form. For this type of system where the raw data is effectively being given away, only an open licence makes sense. This is also the case with some forms of tiled maps where vectors are used to draw the cartography instead of image tiles. The new release of Google Maps in May 2013 introduced WebGL support for rendering maps on the browser. This marks a shift away from tiled maps and towards vector based cartographic content which can be manipulated in a more interactive way by the user. Cartagen[10] is an open source project that pre-dates Google's use of vector mapping to provide a similar set of features. In this case, OpenStreetMap data is used to draw maps on the browser, again using WebGL for graphics acceleration.

[10]Cartagen website: *http://cartagen.org*.

Other tiled map systems worth mentioning are ‘Mapnik’, which is another open source project often used for rendering OpenStreetMap data or making choropleth maps. ‘TileMill’ is an open source map design studio which runs on the desktop and uses Mapnik to render its maps. Its purpose is to make it simple for inexperienced users to create maps. The ‘MapBox’ suite of software tools, including the ‘MapBox GL JS’ javascript library for WebGL maps, is designed around an online maps designer and includes image maps in addition to the vector tile maps used with “MapBox GL JS” library. ‘Leaflet’ is also worth mentioning here, which is designed to be a lightweight javascript library for web-based maps. All of the projects mentioned here perform functions, which, when used together in the correct way, build into a fully-functioning web-based mapping system.

The one feature that almost all web mapping systems have in common is that they expect the data to remain static. With cartographic data, this is generally the case as changes to roads and rivers are relatively infrequent and cache misses are rare, but for applications where the data backing the maps changes routinely, cache misses are high and the requests pass from the edge to the application servers. This causes a problem with the visualisation of volunteered geographic information, or any other real-time system, because the data is constantly changing. This defeats most buffering and caching systems and results in the load being passed on to the application servers as the map tiles need to be constantly re-rendered. One answer to this is to do the map rendering on the client, so, for complex fast changing data, tiled maps may not be the answer. Finally, the one remaining question regarding web mapping is whether a full function web GIS is possible? Google mail, documents, spreadsheets, presentations, calendars and even SolidWorks CAD (OnShape) are now possible through WebGL.

4.3 Static Web Maps

The architecture for static web-based maps is to have all the tiles for the maps pre-rendered on the server as images files, usually of 256 pixels square, and either JPEG or PNG format. This has the advantage of reducing the map to simple HTTP transfers with minimal load on the application servers. The main disadvantage of this system is that the data cannot be changed easily, requiring the re-rendering of possibly millions of tiles.

The main reason that this system works comes from the realisation that either the maximum zoom level or the geographical area of interest can be restricted. Taking global scale data, for example countries coloured by their Gross Domestic Product (GDP), zooming in to street level is pointless as there is only one data value for the whole country. At the other end of the scale, if the data only covered a 200 metre square area, for example from a GPS tracked sensor experiment, then map tiles outside of this area do not need to be created.

Going back to formula 4.7 which was derived in section 4.2 to count the number of tiles required for storage purposes, the effects of the zoom level and area restrictions can be quantified:

$$N = \frac{1 - 4^{z+1}}{-3} \tag{4.15}$$

In equation 4.15, N is the total number of tiles required for all zoom levels up to z. As this progression is growing by 4^z with each zoom level, choosing a low z results in an exponential reduction in the number of tiles for the global scale data restriction.

The second case of restricted geographic area is slightly harder to express, but is based on the fact that only a single tile containing the data is required for all zoom levels until the tile size is smaller than the extents of the data. In other words, the number of tiles follows the progression:

$$N = \underbrace{1 + 1 + 1 + 1 + \ldots}_{\texttt{data extents < tile}} + 4^0 + 4^1 + 4^2 + \ldots + 4^n \tag{4.16}$$

Although this is a simplification as the data is unlikely to fit the tile grid exactly, it serves as a starting point for the analysis[11]. The key is to calculate the zoom level at

[11] London data is a classic case in point as it sits either side of the Greenwich Meridian. Even at the first level of zoom, there are two tiles required because it crosses the first quad-tile boundary.

which the tile size becomes smaller than the data extents and the exponential growth starts (4^0 in equation 4.16 above).

An approximate value can be found by using the Earth's circumference, c, as defined by the WGS84 semi-major axis definition as:

$$a = 6,378,137 \text{ metres} \tag{4.17}$$

$$c = 2\pi a = 40075016.69 \text{ metres} \tag{4.18}$$

Then the width of a tile at any zoom level, z, is given as:

$$W(z) = \frac{c}{2^z} \tag{4.19}$$

Then calculate the zoom level at which $W(z)$ is less than the data extents W_{ex}:

$$2^{z_{ex}} = \frac{c}{w_{ex}} \tag{4.20}$$

$$z_{ex} = floor(\log_2 \frac{c}{w_{ex}}) \tag{4.21}$$

Where $floor(x)$ rounds to the largest integer not greater than x.

The complete formula can then be written as:

$$N = z_{ex} + \frac{1 - 4^{z - z_{ex} + 1}}{-3} \tag{4.22}$$

Using the example above where a 200 metre square of data is to be plotted, $w_{ex} = 200$, giving $Z_{ex} = 17.61$ and total number of tiles as $N = 17 + 85 = 102$.

This data is plotted in figure 4.19, which shows how varying the maximum zoom level and data extents affects the required number of tiles.

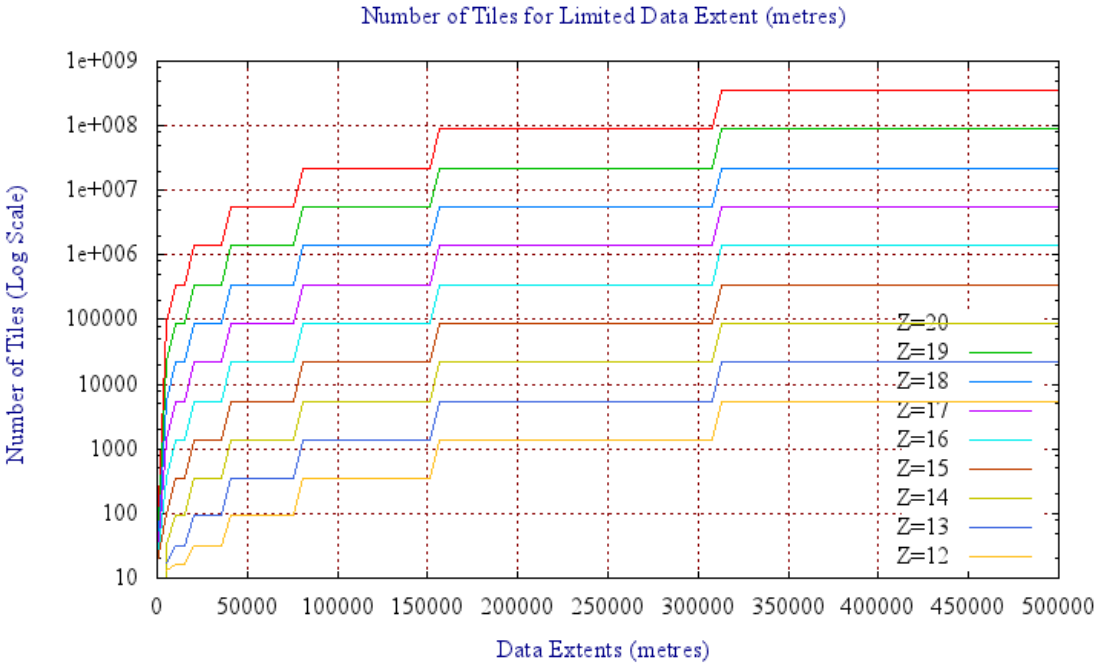


Figure 4.19: Number of static tiles required for different zoom levels and increasing data extents.

4.3.1 Tiling Applications

The graph in figure 4.20 shows the download statistics for CASA's "GMapCreator" application which builds working Google Maps based websites from data contained in shapefiles.

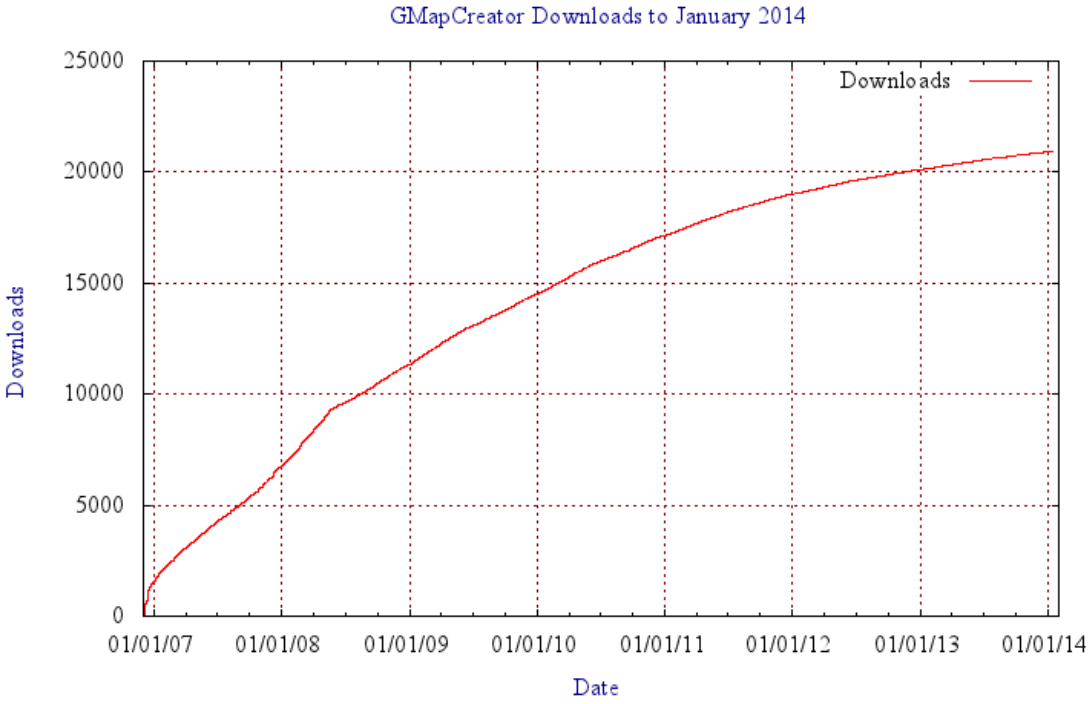


Figure 4.20: Downloads of CASA's GMapCreator tool from December 2006 to January 2014.

The GMapCreator application was built using 'Java' and the 'Geotools' library and was used to create the 'London Profiler' website in addition to being the subject of a number of publications [Hud+07a], [Gib+08], [Hud+08] and [Bat+10a]. One commercial alternative to this is "Global Mapper"[12], released in November 2007 by Blue Marble Geographics which also makes maps from vector data in shapefiles, while "MapTiler"[13] is a comparable tool for raster format data using the GDAL library. What all these applications have in common is that the tile pyramids they create can take a significant amount of time and computing resources to generate. Knowing the number of tiles that will be produced is an essential piece of information that can be the difference between minutes, hours or days. This also highlights the main drawback of this method, namely that the tiles created are static. If the data changes then the entire tile pyramid needs to be recreated. While it is entirely possible to rebuild only the parts of

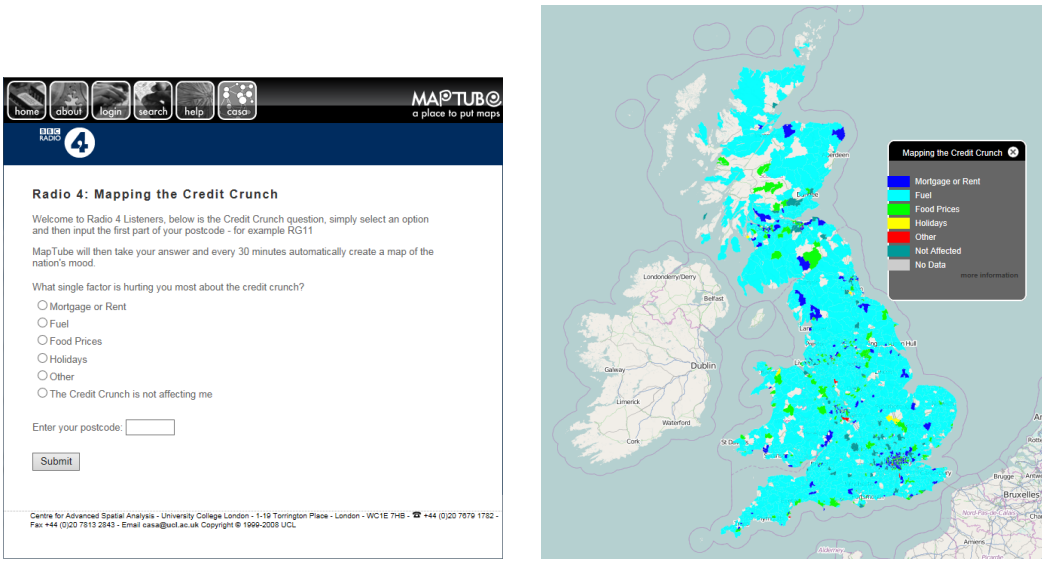
[12]Global Mapper website: *http://www.globalmapper.com*.
[13]MapTiler website: *http://www.maptiler.com*.

the tile pyramid where the data has changed, server-side code is required to propagate the 'change pyramid' through the full tile pyramid based on the geographic bounds of the data that has changed. This could explain the reason which this technique has not received much coverage in the literature, because the tiles at the top of the pyramid would be refreshed with almost *any* data change, and these are the slowest tiles to render as they contain large quantities of data. The technique itself, although seemingly a smart idea, would appear to be flawed. Looking though the code for "Mapnik", "mod_tile" (Apache renderer for OSM data based on Mapnik), "tilecache" (MetaCarta WMS-C server), "tilestash" (Python tile renderer) and "MapSlicer" (Python Open Source), none of these projects implement partial tile pyramid re-rendering, probably because it would require direct coupling between the tiler renderer and the maps database. After reading the online forums and an further background search, the consensus is that a database trigger and custom code to detect the affected envelope and start a tile re-render is the way to go. Finally, another interesting hybrid approach to dynamic choropleth mapping using static tiles is to render all the tiles as PNGs with an indexed colour table, then to use a web service to change the colour table dynamically as the data changes [SD08].

Despite the main use of static tile creation being the ability to create a map using simple HTTP file transfers, it is still possible to achieve user interaction with the map in the form of click-able features and brushing, where moving the mouse over the map draws the user's attention to the data underneath. This can be achieved using 'Data Tiles', where text files are created that match the map image tiles, but which contain a feature ID number for each pixel on the map tile. These feature tiles are downloaded and cached by custom Javascript code on the web page and used to request XML fragments of the data used to make the map at the point the mouse is currently over. These XML files can then be transformed using XSLT into graphs or other infographics related to the map position. If the data on the tiles is encoded using 'run-length encoding'[14], but with a cumulative count of pixel numbers, then the search for a feature ID based on (X,Y) location can be reduced to a binary search. The pixel number is given by $P(X, Y) = X + Y \times 256$. By starting in the middle of the run-length encoded data, the cumulative pixel number is either greater than, less than, or equal to $P(X, Y)$, so the binary search can continue dividing the search space in half until the pixel is found.

Figure 4.21 shows one application of the static tile generation technology. This was

[14]Run-length encoding is a standard compression technique, for further details see: *http://en.wikipedia.org/wiki/Run-length_encoding*.



(a) Input Form (b) Final Map

Figure 4.21: The Radio 4 Credit Crunch 'Mood Map' showing the form used to enter data and the final map of the results [Bat+15] and [GMH15].

a web-based survey conducted on behalf of Radio 4 in April 2008 to determine the public attitude to the 'Credit Crunch'. Further information on this crowd-sourcing experiment can be found in [GMH15]. It consisted of a page on a website where people could select one of the options, put in their postcode and see the result as a choropleth map. This involved aggregating the number of votes for each choice and postcode area, then joining the data with the shapefile containing the postcode districts' geometry and finally running a process to generate $37,651$ tiles covering the UK. The main problem with this work flow is that static tile generation does not cope with fast changing data and people expect to see the data they just entered on the map immediately, not updated every half hour. With this survey, and the others that followed it, spikes in the number of responses are evident following publicity in the media, usually following news bulletins. This equates to a requirement to redrawing the map more than once a second at times of peak load, something which the original technology was incapable of handling.

As far as the work flow required to generate the final map is concerned, the main operations are: "select max column", "join", "reproject" and "tile". Figure 4.22 shows the flow of data through the system.

```
1 java -jar R4CreditCrunch.jar d:\inetpub\databases\R4CreditCrunch.mdb
    England_pcd_1998_area.shp creditcrunch.shp
2 java -jar gmapcreator.jar -dfile=creditcrunch.xml
3 xcopy creditcrunch-tiles\*.* D:\Inetpub\Wwwroot\R4CreditCrunch\creditcrunch-
    tiles /C /Q /Y
```

Listing 4.1: Script file used to recreate the Radio 4 Credit Crunch map.

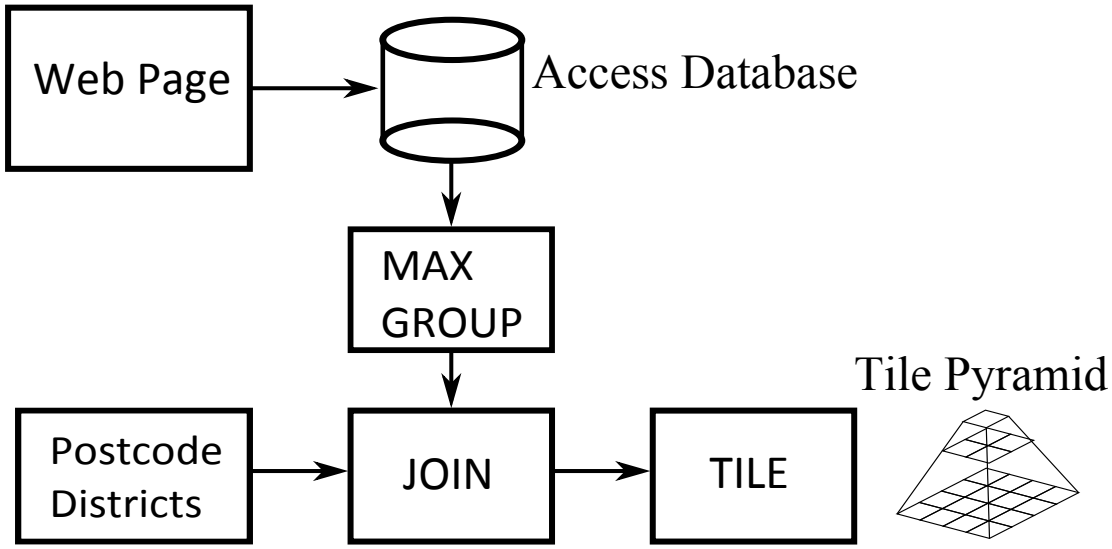
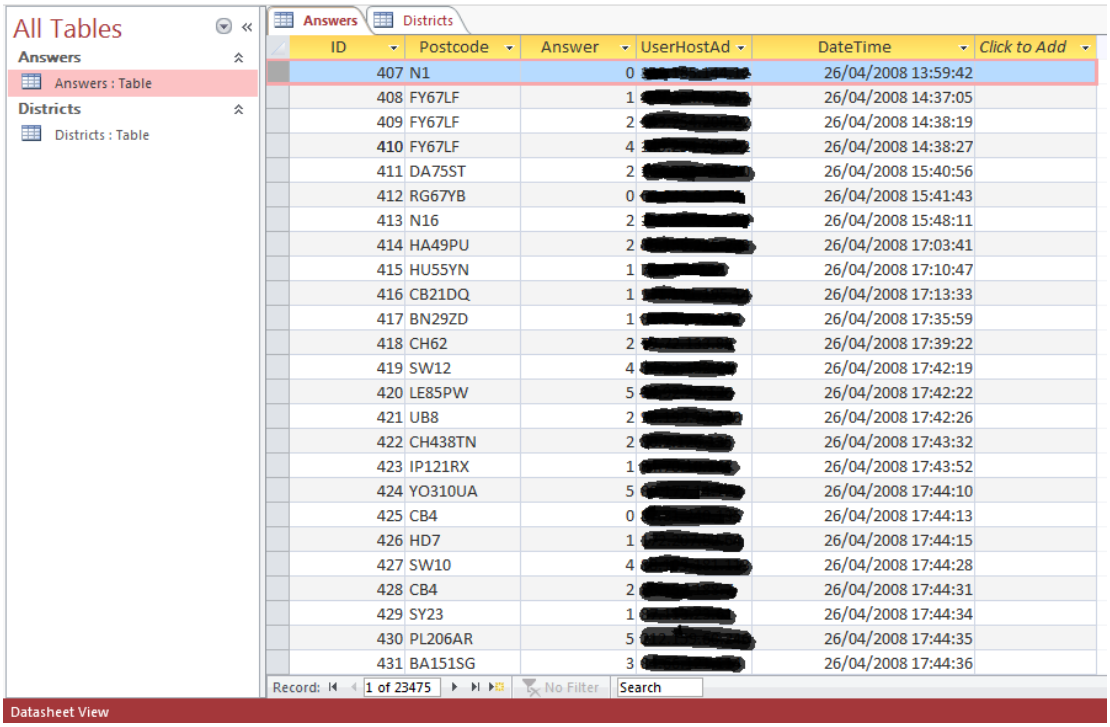


Figure 4.22: Radio Four 'Credit Crunch' system diagram.

In the UK, there are 2651 postcode districts, so there were no performance problems with the data preparation stage. Listing 4.1 shows the work flow used to rebuild the map. First, a Java process connects to the database via JDBC and builds the 'Districts' table (figure 4.24) from the raw 'Answers' table (figure 4.23) using a select which performs a full table scan. The cursor iterates over every row, finds the answer column containing the greatest number of votes and increments the count for that postcode district and response in the 'Districts' table. After the table containing the data to be mapped has been created, another process joins it with the Postcode Districts boundary file and writes out a shapefile ready for the tiling stage.

The tiling operation uses the 'GMapCreator' program, which is also java based and has to reproject the data from OSGB36 into EPSG3857 before starting the quad tree tiling process. This is a disk-bound I/O process as the software has to create 37,651 map tiles of 256 pixels square based on quad tree segmentation of the shapefile. The segmentation algorithm starts with the entire world as a single tile and divides in four each zoom level iteration. Every tile within the data bounding box is rendered as a choropleth with the colours and data ranges specified and then written to disk. The key



All Tables

Answers

Answers : Table

Districts

Districts : Table

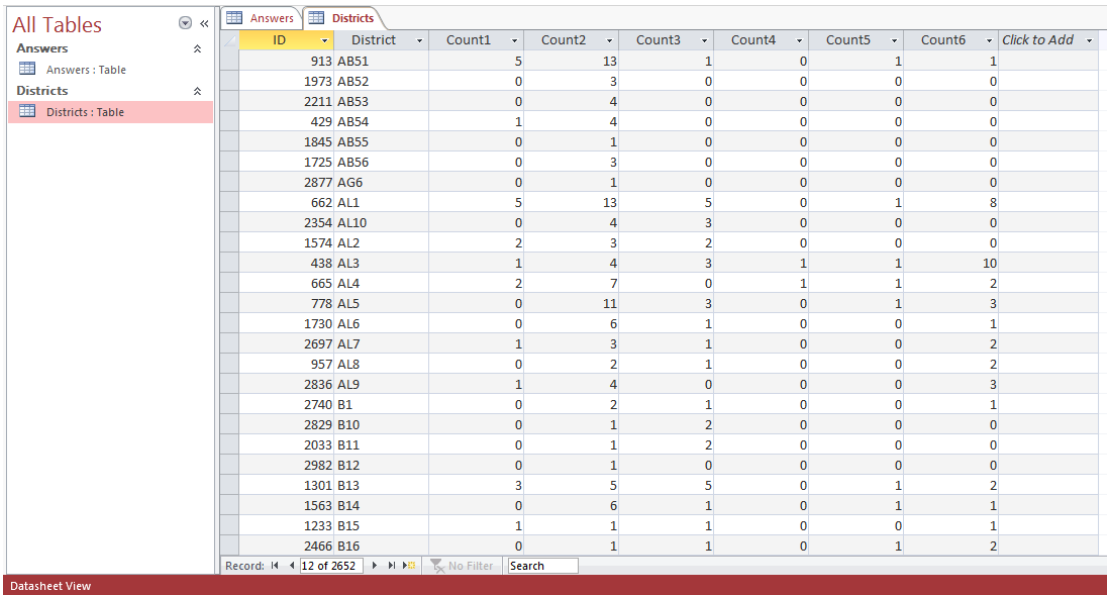
Answers | Districts

ID	Postcode	Answer	UserHostAd	DateTime	Click to Add
407	N1	0	[redacted]	26/04/2008 13:59:42	
408	FY67LF	1	[redacted]	26/04/2008 14:37:05	
409	FY67LF	2	[redacted]	26/04/2008 14:38:19	
410	FY67LF	4	[redacted]	26/04/2008 14:38:27	
411	DA75ST	2	[redacted]	26/04/2008 15:40:56	
412	RG67YB	0	[redacted]	26/04/2008 15:41:43	
413	N16	2	[redacted]	26/04/2008 15:48:11	
414	HA49PU	2	[redacted]	26/04/2008 17:03:41	
415	HU55YN	1	[redacted]	26/04/2008 17:10:47	
416	CB21DQ	1	[redacted]	26/04/2008 17:13:33	
417	BN29ZD	1	[redacted]	26/04/2008 17:35:59	
418	CH62	2	[redacted]	26/04/2008 17:39:22	
419	SW12	4	[redacted]	26/04/2008 17:42:19	
420	LE85PW	5	[redacted]	26/04/2008 17:42:22	
421	UB8	2	[redacted]	26/04/2008 17:42:26	
422	CH438TN	2	[redacted]	26/04/2008 17:43:32	
423	IP121RX	1	[redacted]	26/04/2008 17:43:52	
424	YO310UA	5	[redacted]	26/04/2008 17:44:10	
425	CB4	0	[redacted]	26/04/2008 17:44:13	
426	HD7	1	[redacted]	26/04/2008 17:44:15	
427	SW10	4	[redacted]	26/04/2008 17:44:28	
428	CB4	2	[redacted]	26/04/2008 17:44:31	
429	SY23	1	[redacted]	26/04/2008 17:44:34	
430	PL206AR	5	[redacted]	26/04/2008 17:44:35	
431	BA151SG	3	[redacted]	26/04/2008 17:44:36	

Record: 1 of 23475 No Filter Search

Datasheet View

Figure 4.23: Radio 4 Credit Crunch Answers Table.



All Tables

Answers

Answers : Table

Districts

Districts : Table

Answers | Districts

ID	District	Count1	Count2	Count3	Count4	Count5	Count6	Click to Add
913	AB51	5	13	1	0	1	1	
1973	AB52	0	3	0	0	0	0	
2211	AB53	0	4	0	0	0	0	
429	AB54	1	4	0	0	0	0	
1845	AB55	0	1	0	0	0	0	
1725	AB56	0	3	0	0	0	0	
2877	AG6	0	1	0	0	0	0	
662	AL1	5	13	5	0	1	8	
2354	AL10	0	4	3	0	0	0	
1574	AL2	2	3	2	0	0	0	
438	AL3	1	4	3	1	1	10	
665	AL4	2	7	0	1	1	2	
778	AL5	0	11	3	0	1	3	
1730	AL6	0	6	1	0	0	1	
2697	AL7	1	3	1	0	0	2	
957	AL8	0	2	1	0	0	2	
2836	AL9	1	4	0	0	0	3	
2740	B1	0	2	1	0	0	1	
2829	B10	0	1	2	0	0	0	
2033	B11	0	1	2	0	0	0	
2982	B12	0	1	0	0	0	0	
1301	B13	3	5	5	0	1	2	
1563	B14	0	6	1	0	1	1	
1233	B15	1	1	1	0	0	1	
2466	B16	0	1	1	0	1	2	

Record: 12 of 2652 No Filter Search

Datasheet View

Figure 4.24: Radio 4 Credit Crunch Districts Table generated from the responses in figure 4.23.

to optimising this type of tiling algorithm is in the spatial indexing used to select only the data on the tile which is currently being drawn. As for writing the tiles to disk, the number required is fixed, so there are always going to be 37,651 tiles compressed using JPEG compression and physically written to disk. Algorithm 2 shows how the quad tree tiling works. In practice, there is a concurrency problem that occurs when users are accessing the map at the point when a new one is being created. This issue is solved by creating the new map tiles in a different directory and swapping between the old one for the new one using by using a directory rename command ('xcopy' in listing 4.1). While this is not guaranteed to be indivisible, it is adequate for a web-based map.

Algorithm 2 Quad tree

Require: $Features$, the data to draw on the map
Require: $TileCodeString$, code string which uniquely identifies the current tile
Require: $TileExtents$, the geographical extents of the current tile
Require: $MaxZoom$, final zoom level of required tile pyramid
1: $Z \leftarrow length(TileCodeString)$ {$TileCodeString$ increases by one char every zoom level}
2: **if** $Z > MaxZoom$ **or not** $TileExtents$ covers($Features$ bounds) **then**
3: RETURN {Guard Case for depth or outside data bounds}
4: **end if**
5: $Features \leftarrow$ `All features within` $TileExtents$ `box`
6: **for all** Features **do**
7: Render feature on current tile using $TileExtents$ for translation and scale
8: **end for**
9: Write tile to disk, named "$TileCodeString$.png"
10: $SplitX \leftarrow (TileExtents.MinX + TileExtents.MaxX)/2$
11: $SplitY \leftarrow (TileExtents.MinY + TileExtents.MaxY)/2$
12: Split TileExtents into 4 quadrants: SW, NW, NE and SE based on SplitX and SplitY
13: Quadtree($TileCode$+"t", SW) {Southwest}
14: Quadtree($TileCode$+"q", NW) {Northwest}
15: Quadtree($TileCode$+"r", NE) {Northeast}
16: Quadtree($TileCode$+"s", SE) {Southeast}
17: **return**

Figure 4.25 shows a cumulative plot of the total number of responses to the survey over time. From this data it is easy to see that the majority of responses came on the 17th of May, when the survey was publicised on the BBC website.

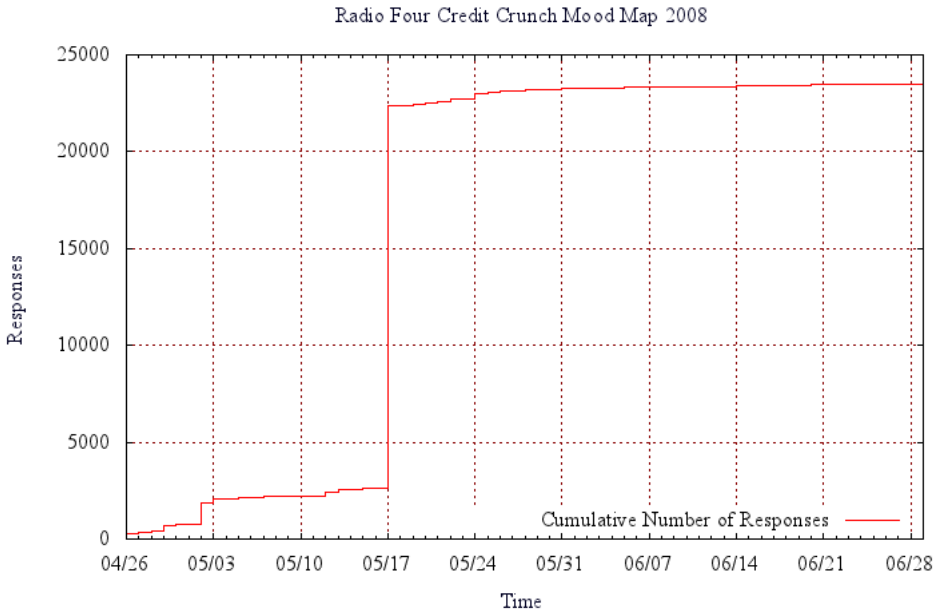


Figure 4.25: Radio 4 Credit Crunch Responses plotted as a cumulative graph of response numbers over the period of the survey.

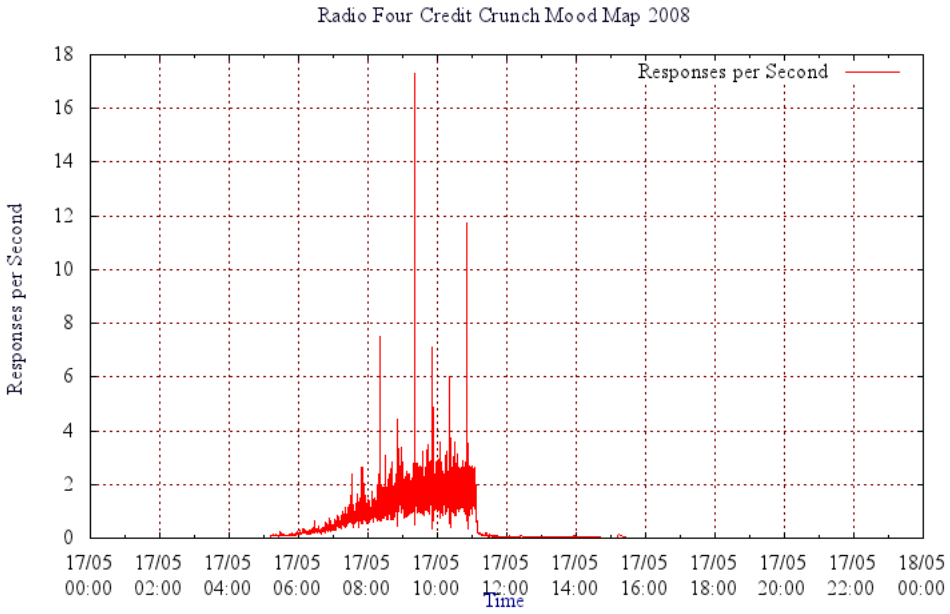


Figure 4.26: Radio 4 Credit Crunch Responses per second for 17 May 2008. Responses per second is calculated using a sliding window of the time for the last 20 responses: $\frac{20}{T(n)-T(n-20)}$

Figure 4.26 shows a more detailed picture for the 17th May where it can be seen that the automated map was taking almost 18 responses per second at its peak. Using this architecture, the map building process is de-coupled from the response input, so, as long as the web input form and database can handle data at this speed, the map will always update every half an hour. If the map was being redrawn every time a new

response was entered, then the architecture would need to cope with 18 map redraws per second.

In summary, the benefits of static tile pyramid creation is the simplicity and ability to scale to loads of 1000 hits per second, while at the same time circumventing any legal issues with the dissemination of the map's vector data. According to the performance tests on Microsoft's IIS7 web server, this architecture running on a quad core Server 2008 SP2 machine is capable of serving 5,000 tiles per second (see: *http://www.webperformance.com/load-testing/blog/2011/what-is-the-fastest-webserver/*). The disadvantages are in the handling of fast-changing data, or exploratory data analysis where a map might be created and viewed only once. Creating static tile sets is for publishing static data to a wide audience because of its scalability. Building tile pyramids on demand with fast changing, dynamic data is not an option. In addition to this, further exploration of the data is a desirable quality as it is possible to draw individual maps of the spatial density of each response which would multiply the map building load.

4.4 Dynamic Maps

Making a map from a dataset involves applying spatial context to the data, which is then styled to produce the final visualisation. In this sense, spatial data transformed into a map can be seen as analogous to XML transformed into XHTML and displayed in a browser using CSS rules. 'MapCSS'[15] is an extension of the CSS standard applied to the styling of map features, while the OGC standards 'Styled Layer Descriptor' and 'Symbology Encoding' perform similar functions and are supported by many GIS packages.

$$\texttt{Map} = f(\texttt{Data}, \texttt{Spatial Context})$$

In semantic and linked data, unique URIs are used to identify entities. An example of this is the UK Ordnance Survey's linked data on the UK, where geographic features are defined using URI identifiers.

By extending this idea, spatial data can be placed at an accessible location on the web and its URI passed to a web service, or other application, for rendering on a map. This is a subtle variation on the way tile servers like 'Mapnik' work, but it has important consequences. Systems like 'Mapnik', 'Geoserver' and 'TileServer' all expect their data to be stored locally, usually in geodatabases like ArcGIS and PostgreSQL, or stored on the local file system as shapefiles. The reason for this is very simple, as geographic data can often be very large, giving rise to excessive wire time in transferring the data and unacceptable delays for the user. Taking the ONS Census Boundaries as an example, the OA level data for the 2011 census containing only the boundary data is 932.3MB (the generalised 20m version of this is only 69.6MB though). This would take 372.92 seconds (6 minutes) to transfer on a 20 MBit link (the generalised file would still take 27 seconds). By contrast, the aspatial population data (KS101EW) for the same Census is in CSV file format and only 17.5MB in size including 12 separate population variables, taking 7s to transfer on a 20 MBit link.

The solution is to specify the aspatial data with a URI, but include information on the spatial data to join it with. The map making process now becomes a work flow where the aspatial data is loaded from the URI, joined with a spatial dataset held on the server, using a primary key column, then styled before the map is drawn. The aspatial data is small and unique, while the spatial data is large and shared between many datasets, so it makes sense to hold spatial data local to where the rendering is performed.

[15] MapCSS definition: *http://wiki.openstreetmap.org/wiki/MapCSS*.

The remainder of this section details the evolution of this design as part of the Genesis project [CC08], resulting in the National e-Infrastructure for Social Science (NeISS) project to implement web-based mapping of the 2001 Census and the SurveyMapper project ([GMH15] and [Bat+15]) to implement web based geographic surveys which improve on the 'Mood Maps' from the last section. The result of this development is a dynamic tile renderer capable of handling millions of hits per hour. The various strengths and weaknesses of this approach are then examined with a view to building a next generation mapping system in the final section.

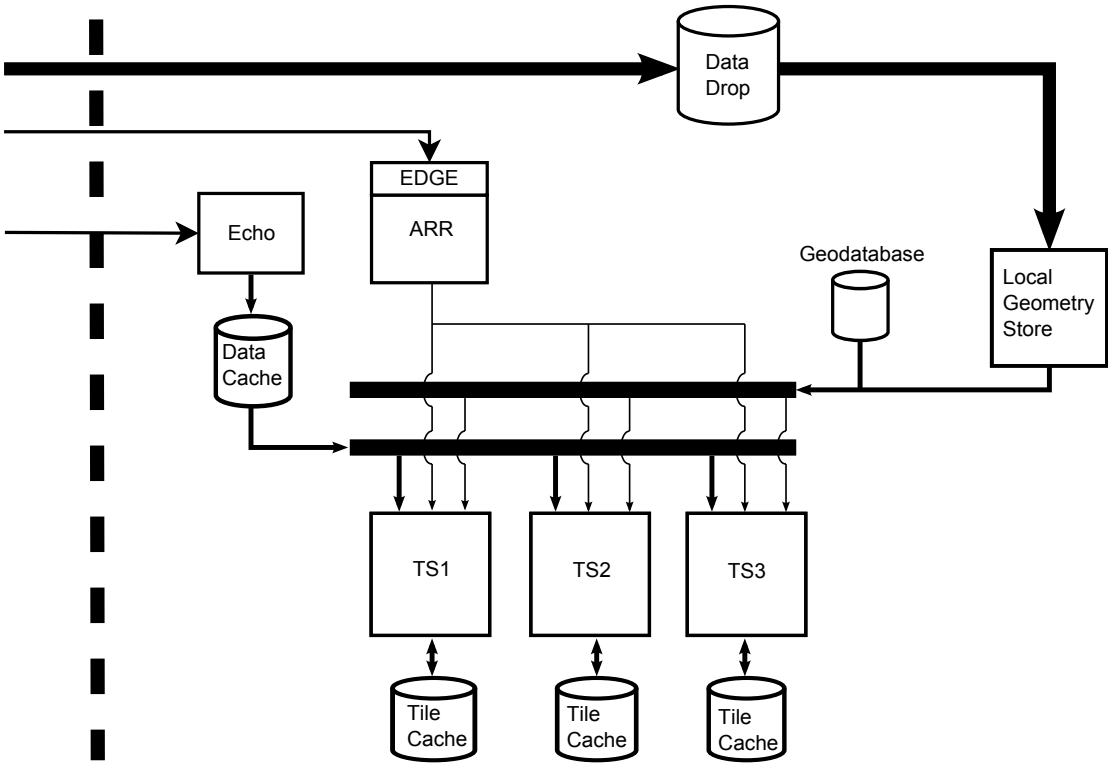


Figure 4.27: MapTubeD dynamic map tiler system diagram.

Figure 4.27 shows the architecture resulting from this design idea. A key feature is that the system must be scalable by adding additional processing elements as required. This, coupled with the multi-threaded asynchronous nature of tile rendering, leads to a design that needs to be able to buffer and cache data intelligently.

The first problem is a very simple one and affects all web servers when rendering map tiles. A Google Map, or OpenLayers map, when first opened on a web page, makes an immediate request for between 30 and 40 map tiles dependent on the window size. These arrive at the tile server asynchronously, but, for new data not yet loaded, there will be a delay while the aspatial data is pulled from its URI. Rather than having

30 threads all load the same data almost simultaneously, the loading is handled through a singleton data cache object which blocks all threads for data until it has loaded. Early versions of the software blocked all rendering threads until new data had loaded, but this was replaced with a more intelligent data loader following problems with bad HTTP requests blocking all tile rendering until the request timed out.

Once data is available, the rendering follows a similar pattern to the static tile rendering outlined earlier, except that the spatial data now comes from a SQL Server database. The concept of 'late binding' is used, where the aspatial and spatial data is only joined at the point where the rendering is about to happen. Reprojection of the data is not necessary as this has been done in the pre-processing step when it was first loaded into the database. In addition to this, extensive use is made of multiple levels of detail (LOD)[16] using simplification of the base geography. This is done when a new dataset is loaded and prepared, along with generating a spatial index.

The levels of detail were chosen following performance tests on rendering speed, with the reduction levels calculated based on the size of a pixel at the respective level of zoom (5, 10 or 13). This requires a 'Metres per Pixel' value (m), which can be calculated from the world width at the equator (WW) and zoom level (z) as follows:

$$m = \frac{WW}{256 \times 2^z} \tag{4.23}$$

Using the WGS84 semi-major constant (a), the world width in metres as used in the spheroid is calculated as follows:

$$WW = 2\pi a = 2 \times \pi \times 6378137 = 40075016.68 \tag{4.24}$$

Substituting this value for WW into equation 4.23, the decimation limit for zoom level 5 can be calculated:

$$m = \frac{40075016.68}{256 \times 2^5} = 4891.97 \texttt{ metres per pixel} \tag{4.25}$$

[16]Level of detail (LOD) and decimation are techniques for reducing the number of points in the data. Decimation works by fitting the X and Y values to a regular grid based on the pixel size of the final output so that two points which map to the same pixel are collapsed into a single point in the simplified data.

Obviously, this value is latitude dependent, but the aim is to obtain upper and lower limits for the optimisation process. In light of this, it should also be noted that if both the X and Y coordinates are rounded to this 1 pixel amount, then the total distance moved is actually $\sqrt{1^2 + 1^2} = \sqrt{2}$ pixels. In practice then, a 0.5 pixel error is more acceptable visually, but the exact values used can be optimised based on the latitude of the data bounds.

While level of detail is useful in reducing the number of points that need to be rendered and so increase the rendering speed, it does nothing to reduce the number of features. If a map had 34,000 separate features to be drawn at the maximum zoom level, then, even with decimation, there are still 34,000 features at zoom level 5. At the higher level of zoom, spatial indexing comes into play, so only the features on the map tile currently being drawn are handled. Unfortunately, this system does not work at lower zoom levels when the entire data bounds fit on a single tile. Each feature may be reduced to a single pixel in size, but there can still be a large number of features which need to be drawn correctly if the map is not to have holes in it.

Fortunately, this is where tile caching comes in. There are comparatively few tiles containing all the data[17], so the approach taken is to cache these top level tiles, giving them a high value to prevent them from being deleted. The cache expiry scheme is 'Least Recently Used' (LRU), but with high zoom level tiles deleted first.

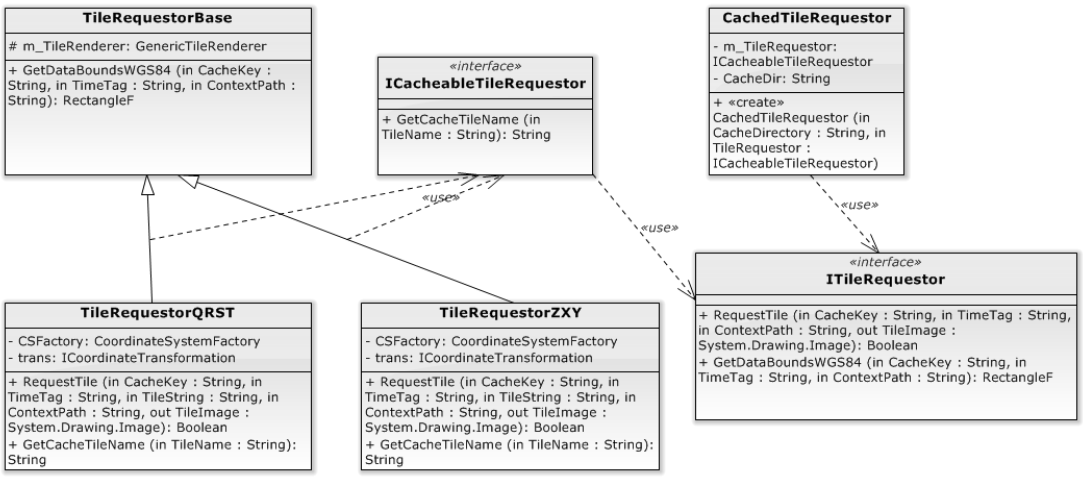


Figure 4.28: UML Class Diagram for the MapTubeD TileRequestor.

Figure 4.28 shows the MapTubeD 'TileRequestor' class diagram. The MapTubeD library contains a 'TileRequestor' class which has specialisations for the two most com-

[17] Recall the analysis in section 4.3 on static web maps and data bounds.

mon methods of tile naming: the Keyhole ‘qrst’ format used in Google Maps versions 1 and 2, and the ‘Z_X_Y’ format used later by Google and defined by the OGC as the WMTS standard. A ‘CachedTileRequestor’ owns a plain ‘TileRequestor’ and provides additional methods for managing a disk-based tile cache. The pattern used is for the ‘CachedTileRequestor’ to attempt to satisfy a tile request from the cache and only pass on the request to the full ‘TileRequestor’ to render a new tile in the event of a miss. The reason for the ‘TileRequestor’ returning a boolean value is to avoid caching failed tile requests in the case of exceptions. The ‘CacheKey’ is a plain text sanitised version of the URI of the data being mapped. The cache stores tiles for each map in unique folders based on the URI.

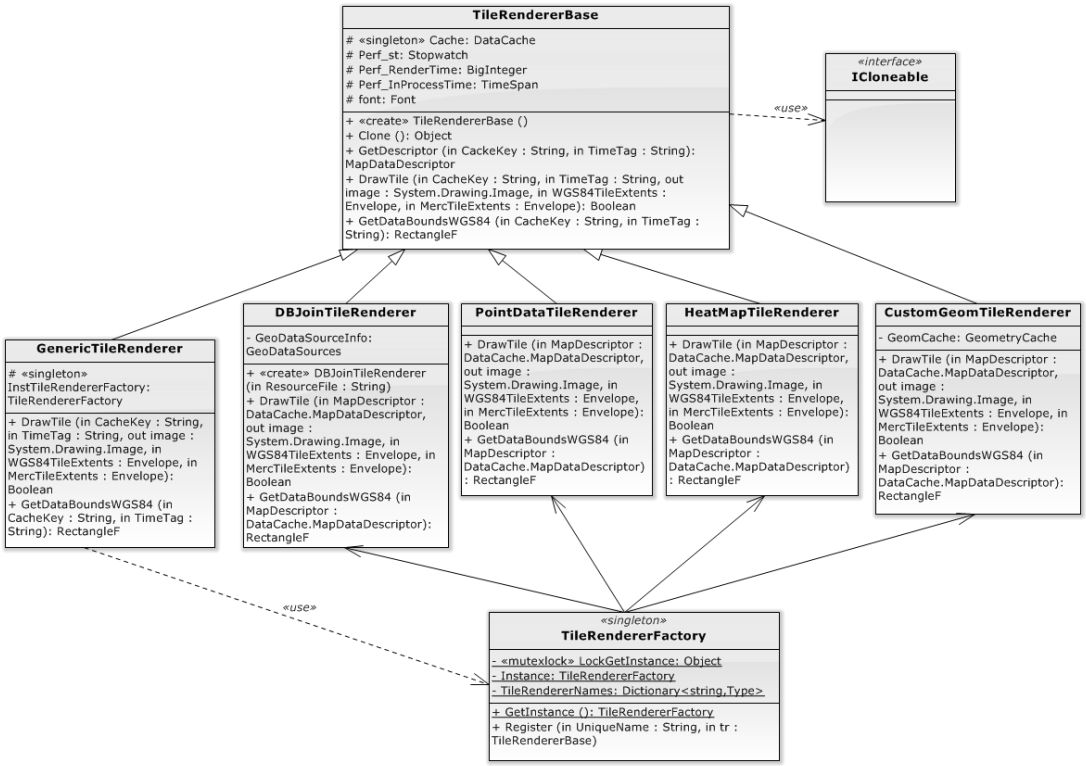


Figure 4.29: UML Class Diagram for the MapTubeD TileRenderer.

The actual tile rendering is achieved using specialised tile renderers for the different types of data. In figure 4.29 the class diagram for the tile renderer architecture is shown. The ‘DBJoinTileRenderer’ is designed to handle CSV data that needs to be late joined with boundary files in the server geodatabase at the point of rendering. The ‘PointDataTileRenderer’ and ‘HeatMapTileRenderer’ both operate on point data, but display symbols and a colour filled heat map respectively. The ‘CustomGeomTileRenderer’ handles user-defined custom geometry which needs to be uploaded to the server,

for example data originating in shapefiles. The 'GenericTileRenderer' is a necessity as all the functionality behind loading and interpreting the data is built into the tile renderer base class. When the 'GenericTileRenderer' has interpreted the data it uses the 'TileRendererFactory' to create an instance of the correct tile renderer for the data type and visualisation.

One problem with this type of tile caching comes with data that is constantly changing and with load balancing. All the processing elements in figure 4.27 need to operate individually to give 100% parallelism, so the idea of global cache messages to cause the cache to flush when new data was available was rejected. This system was employed early on, with the SurveyMapper website sending a message to the tile renderer to flush the cache every time a user added a new response to a survey and the data changed. At the time, only one tile rendering element was being used, so the system worked in a limited sense. If load balancing with multiple tile servers is implemented, however, the flush message then has to go to every tile server in the set, potentially on machines in different physical locations. To avoid this undesirable situation, an additional parameter was added to the tile request, so tiles were then rendered for a specified point in time. This has implications for interleaved tile requests, as multiple copies of the same map at different points in time can now be handled in parallel. This design came from the extreme load placed on the tile servers by high hit rates on the dynamically rendered maps, which is outlined in the following sections.

The UML deployment diagram shown in figure 4.30 shows the final solution for dynamic tile rendering. Multiple tile servers are deployed as virtual machines on a cluster of virtual host servers, forming a tile render farm. The advantage here is that tile renderer servers can be added or removed to load-balance as required. At present there is only a single geodatabase server holding the boundary files, but this could also be load-balanced if required. The structure of this system allows for testing a number of different performance enhancements. Firstly, each tile renderer can be configured to have its own private cache, or to share a cache between all the renderers. Coupled to the tile caching scheme is how the application request routing for the render farm is configured. The settings allow all tiles from a single client to be routed to the same tile renderer, or for them to be distributed evenly across all the servers. The effect of using a single renderer is to concentrate all the tile caching more efficiently, while spreading requests across all tile renderers improves speed, but requires each to have its own copy

of the data to render from. Finally, the edge caching speed and size can be configured, so tiles can be set to render at a speed of no more than every 30 seconds as a final safe-guard against excessive load.

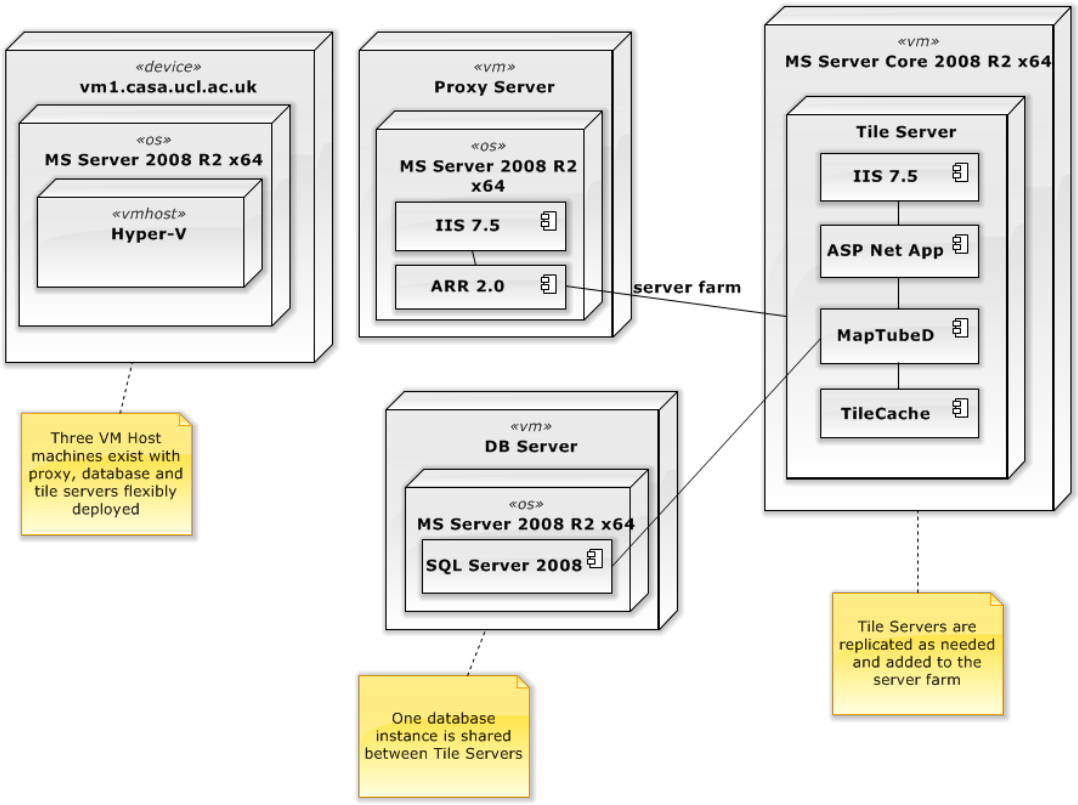


Figure 4.30: UML Deployment Diagram for MapTubeD.

Dynamic tile rendering allows the 'Mood Map' surveys, which were introduced in section 4.3, to change in real time as the data changes. In other words, as people enter responses on the map, they can be taken directly to a map which is rendered on the fly showing the value they just added.

In February 2011 CASA ran a broadband speed survey in conjunction with BBC Look East. Without any prior warning, they started advertising the survey on the daily news bulletins, which resulted in very large numbers of hits. This was prior to the full implementation of load balancing, so everything was run from a single 4 processor 2.66GHz virtualised IIS 7.5 instance with a similarly sized SQL Server 2008 R2 database server. Once it was realised what was happening, the virtualised tile server and database were increased to the maximum limit of the hardware. Two separate virtualisation hosts were being used, both running Microsoft Server 2008 R2 Datacenter and Hyper-V. Full load balancing had not been implemented at this point, so this was a

test of a single tile server under extreme load.

The statistics show that, at the peak load just after the 6pm news bulletin, the single tile server was handling over 200,000 hits in half an hour (figures 4.31 and 4.32).

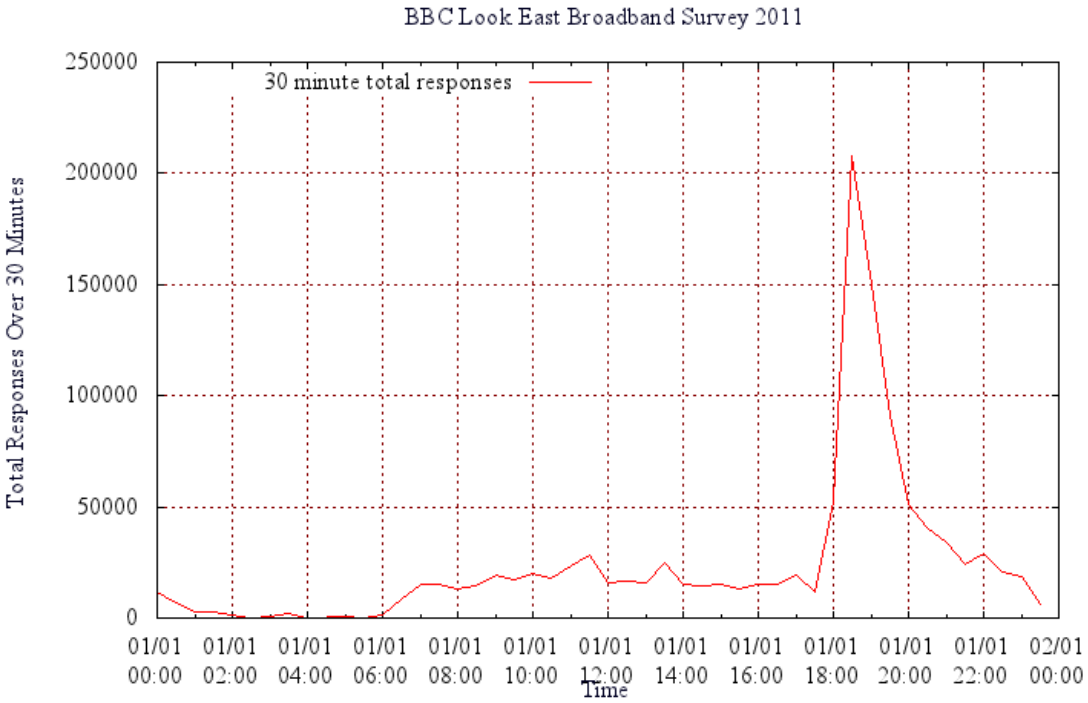


Figure 4.31: BBC Look East Broadband survey, tile server statistics for 22 February 2011. The figures show tile server requests made over a 24 hour period aggregated by half an hour. The peak just after 18:00 shows that over 200,000 responses were recorded in that single 30 minute period.

While this is an impressive test of a single tile server, for full scalability more advanced load balancing is required. By switching the tile requests to specify a point in time, one user can be looking at his map, while a second user, who added new data after the first, can be looking at a newer version of the map in parallel with the first. In addition to this interleaving of map data, further tile servers can be added in parallel as required by the current load. By virtualising tile servers, they can simply be switched on and added to the balance set as required, resulting in flexible use of the available resources. Figure 4.27 shows this system with three parallel tile servers, each with an individual tile cache on the local file system and a single geodatabase. The aim behind designing this system was to make it as flexible as possible, so the tile servers can be reconfigured to share a tile cache, run without any tile cache, or even use their own local geodatabases rather than share a single one. While the idea of running a tile server without a tile cache might seem counter-intuitive, a further improvement to the design has been to employ a technique known as “Edge Computing”. In all web applications

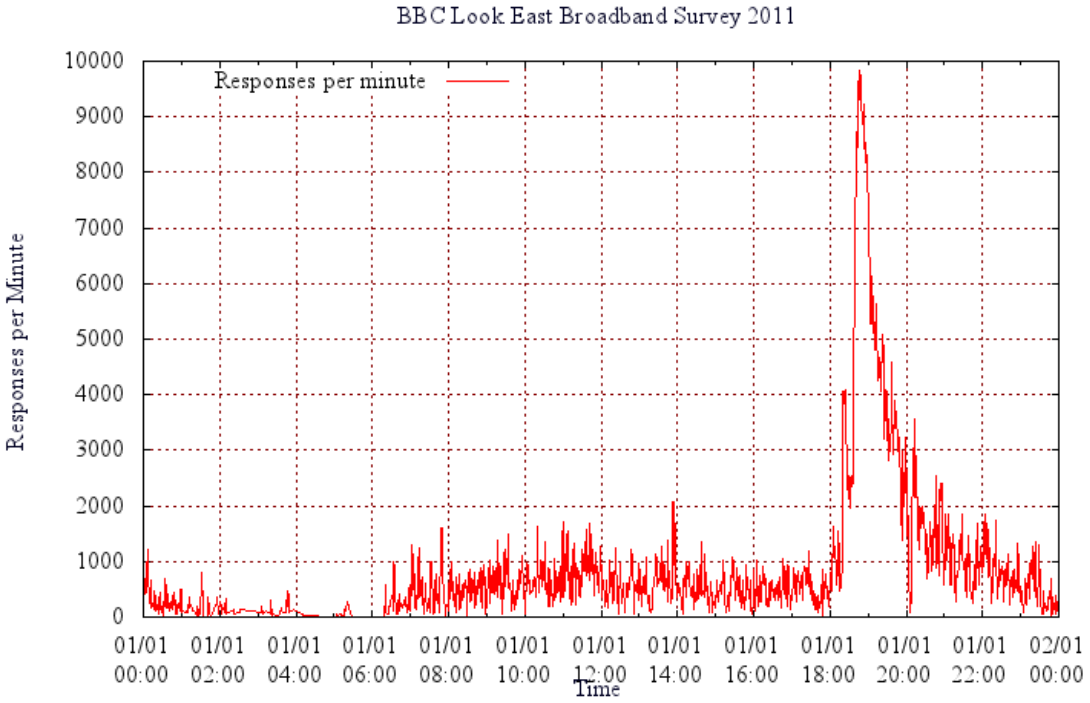


Figure 4.32: BBC Look East Broadband survey, tile server statistics for 22 February 2011. The figures show tile server requests made over a 24 hour period aggregated by minutes.

capable of handling high loads, the aim of edge computing is to satisfy the majority of requests at the edge of the network, thus preventing requests from reaching the application servers unless absolutely necessary. This is achieved using a proxy server[18] with a cache at the edge of the network. Only if the request can't be fulfilled by the edge cache will it then be passed on to the tile server array, where the local cache will be checked, resulting in a tile render for any cache misses. In addition to this, the proxy server can be configured to always route requests from the same source to a particular tile server. This can be advantageous with local tile server caches as it increases the chance of a hit, but if every request is a miss, then the result is a single tile server sequentially rendering all the requested tiles. In practice, the edge cache is used to set a limit on how fast the data can change. By setting a cache interval time of 30 seconds, the edge cache can be used to prevent the maps changing more than once every 30 seconds, reducing the load on the application servers.

[18] In this system the proxy server is Microsoft's Application Request Routing (ARR) version 2.0.

One final problem that had to be overcome with this system is the fact that the raw data is required by every tile server involved in the map rendering. Imagine the case with four parallel tile servers and a request to map a new dataset stored at a remote URI. This results in all of the tile servers trying to load the data simultaneously, so a reverse proxy has been added, as shown by the 'echo' server element in figure 4.27. All the tile servers in the local group are configured to use this as a proxy server, which speeds up access to remote data and introduces a further layer of caching for data being loaded by the system. Although this does create a single point of failure, the reliability to date has been sufficient to avoid the necessity of a backup proxy server to take over if the primary failed.

4.4.1 Automatic Map Generation

While it is easy to dismiss web-based mapping systems as simply fulfilling the requirement to draw tiled maps, looking at the bigger picture, this could be seen as the graphics layer of a conventional GIS system. In designing a scalable web-GIS, the ability to render data automatically from URIs is a useful feature. While this system has some similarities with Google's "Fusion Tables", which was released about a year after this system was first published, there are some important differences. The aim from the start was to build a dynamic system and not focus on the 'load/store' architecture favoured by Google's offering. In the 'load/store' architecture, geospatial data has to be first uploaded to the cloud before the map can be drawn. In the MapTubeD system, it was designed to dynamically pull in datasets from the Internet as required. Each system has its relative strengths and weaknesses, with custom geometry being the Achilles' heel of the MapTubeD system, although it can handle free-form geospatial data as can be seen from the UML diagram in figure 4.29. Where the benefit of this system really pays off though, is when data stores are considered.

Data stores like "*london.gov.uk*", "*data.gov.uk*" and the 2011 Census Release (*https://www.nomisweb.co.uk/census/2011/bulk/r2_2*) contain large quantities of data which are often on the fixed geographies already loaded into the MapTubeD geodatabase. After researching systems like ESRI Marketplace and Geocommons, which enable users to upload geographic data and make maps, it was decided to take a different approach with MapTube.

A work flow was created which allows users to make maps directly from data in CSV files as follows:

1. Either store the CSV file on MapTube, or allow URI input directly
2. Use data mining techniques to identify the spatial context of the data and a column to map
3. Implement a breaks algorithm on the identified data column and choose a colour scale for mapping
4. Map the data using the MapTubeD tile renderer

The concept was to make the map making process easy enough that it could be done using simple point and click. The most complicated part of this work flow is in step 2, where the spatial context of the data is identified. In this system, the data is limited to point data in the WGS84 or OSGB36 coordinate systems, or polygon data identified by a recognisable spatial key. In practice, probabilities are computed for each column belonging to each spatial data set known by MapTube, based on the percentage of matched keys. Heuristics are used in order to speed up the process, where a column needs to match by a threshold amount before being considered as a match.

The process of identifying data in a CSV file for automatic mapping is shown in the UML activity diagram in figure 4.33. This shows the route taken to one of two choices: “Choose Geometry” for polygon boundary data (choropleth map), or “Choose Points” for point data (symbols or heat map). In the case of polygon boundary data, it should be recognised that the boundary data is never available to the user in this system, which can be advantageous when the data is covered by legal restrictions. This was originally the case with the 2001 Census data, but the legal restrictions have been eased considerably for the 2011 release of the Census. The identification stage here simply tags the data as using as particular boundary file and the spatial join happens at the point where the tile server performs the actual rendering of the data on the tile.

One final point worth mentioning is the decision to handle files identified by their URLs. This is a departure from how Google Fusion tables works, as it gives MapTube the ability to map data directly from documents stored on the Internet. It is the separation between the spatial and aspatial data that gives MapTube this ability, allowing it to download relatively small CSV files containing data that are then ‘late-joined’ at the point of rendering.

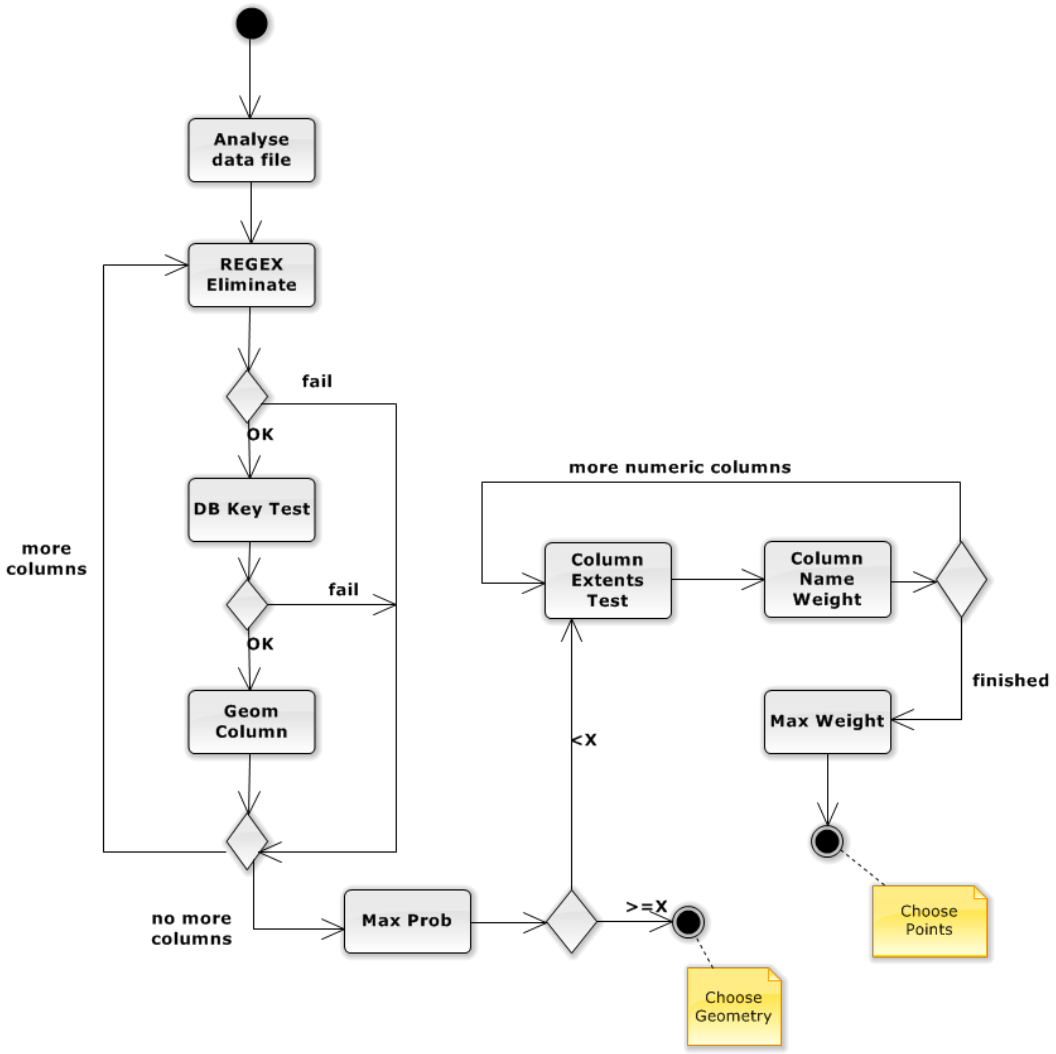


Figure 4.33: UML Activity Diagram for the MapTubeD GeometryFinder.

Identification proceeds by taking the top 100,000 rows in the data and labelling each column according to whether it is identified as numeric, sequential, or text. The full list of features is listed in table 4.1, which includes some additional statistical data used in the later stages of the automatic mapping process. The identification process is essentially a form of data mining with the aim to find spatial context and semantics describing all the columns in the data file.

The '*IsIndex*' predicate was added to prevent the system automatically mapping the feature ID column (FID), which is common in geographic data. The '*MissingDataValue*' is based on the partition of the column values into two sets, one containing only numeric data and the other containing a set of identical '*MissingDataValue*' alpha-

Column Properties Used for Automatic Identification of Geospatial Data

Data Property	Definition
IsNumeric	True if a column contains only numeric data or MissingDataValue alpha symbols
IsIndex	True if a column has a recognisable arithmetic progression e.g. 1,2,3 or 5,10,15 or 10,9,8
Min	Minimum data value for a numeric column
Max	Maximum data value for a numeric column
MissingDataValue	Recurring alpha numeric value detected in numeric column e.g. ‘n/a’ or “” or ‘???’ or ‘-’

Table 4.1: Column Properties used for Automatic Identification.

numeric identifiers. The logic used is to find a single alpha-numeric label which, when removed from the original set, leaves a pure numeric column. This does not allow for ‘no data’ labels which are numeric though, for example ‘-1’ in a column of Cardinals, but it is possible to extract this type of information from the frequency of occurrence and probability density function. Figure 4.34 shows the UML Class Diagram for the MapTubeD “GeometryFinder” class which is used to identify CSV data identified by its URI.

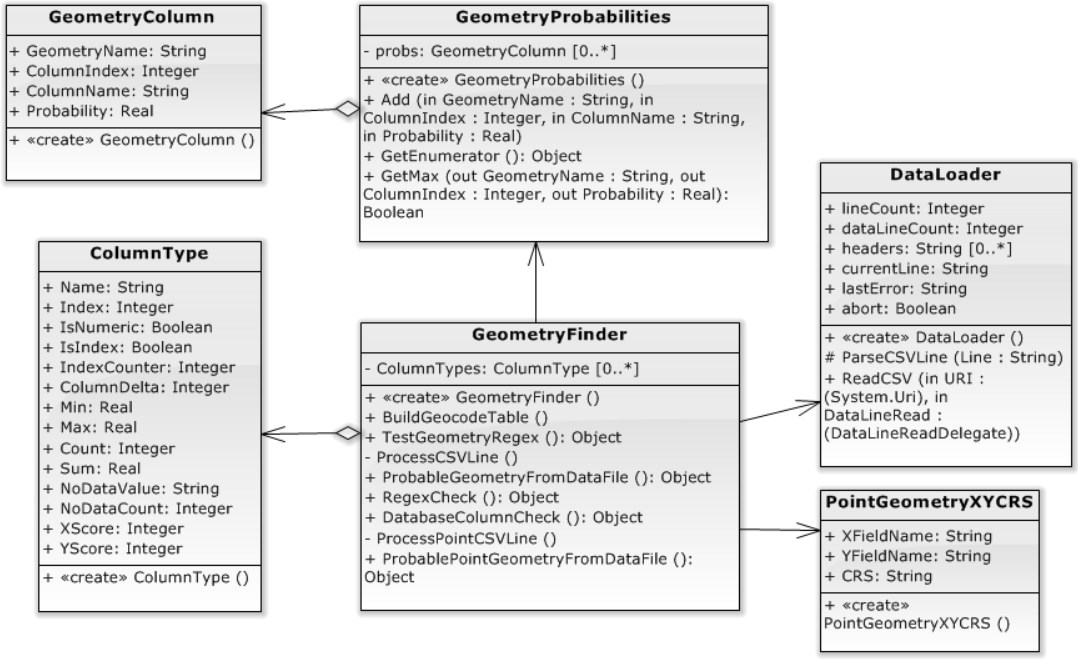


Figure 4.34: UML Class Diagram for the MapTubeD GeometryFinder.

The potential for every column to be an area key field is tested by comparing every row of every column with a regular expression describing a valid key code for every dataset in the server’s geodatabase. This is a heuristic designed to quickly eliminate any columns that could not possibly match any area key in the known geographic datasets

on the server. If the regular expression test returns no match for a column, then that column is discarded. Despite the fact that the regular expression test needs to be performed "$DataColumns \times DataRows \times ServerDatasets$" times, the test that follows is more rigorous and computationally expensive, so any early discards are beneficial to performance. Any columns that survive the first heuristic are then tested against a geocode table on the server. This is a flat table containing a triple of area key string, public dataset name and private table name. The reason for using a public alias for the actual database table names is for security, to prevent names of server tables from appearing in any Internet packets. Columns are matched in chunks against values in the geocode table, with the number of matches used to determine a probability of match against a specific server dataset. Probability is defined as the number of matching rows divided by the total number of rows tested (equation 4.26).

$$P(match) = \frac{N(match)}{N(tested)} \tag{4.26}$$

Every column has a list of zero or more probabilities of match for any of the server datasets. In other words, column A in the data might match server dataset 1 with a probability of 0.9, but it might also match server dataset 2 with a probability of 0.95. The column and server dataset pair with the highest probability of match is chosen as the area key, although, in the case of a low probability match the match probability calculated for point data is also taken into account, which might result in the dataset being labelled as containing point data rather than area data. Where the area data is concerned, the match should be close to 100% as any rows in the csv file that do not match are not going to display on the map. Warnings are generated for any non-matched rows, giving the user the option of overriding the automatic detection or correcting the data. Problems can occur when there are two datasets with very similar area keys, for example the 2001 and 2011 Census data. The keys are identical apart from a very small number of changes between the two Census areas, so a large number of rows need to be tested to ensure the correct geometry is chosen.

Point data detection is based on finding two numeric columns that fit the bounding boxes of either WGS84 coordinates or OSGB36. Weighted heuristics are used to increase the probability that a column will be detected as an ordinate value for point data based on a plain text match with any of the following terms:

```
lat, latitude, lon, lng, long, longitude,
east, north, west, south, x, y
```

Detection of the coordinate reference system is based on the bounding box of the two ordinates. If any points fall outside of the (-180,-90), (180,90) bounds, but within (0,0), (700000,1300000) then it is assumed to be OSGB36.

After the initial processing stage to identify the type of data, the next problem is in determining a suitable colour scale. At this point, the column containing the data to be mapped has been decided, so further analysis of the distribution of data values in this column is performed. The data could be either discrete or continuous, which is decided based on the number of unique data values found in the sample. A cut-off of 12 unique values is used to choose between discrete or continuous, based on the colour scales of [BHH03] having a maximum of 12. For continuous data, breaks are chosen, initially using 5 breaks based on [Fis58]. The number of breaks, and whether they are uniform, natural breaks or quantiles is user configurable after the initial stage. The result of all this work is that the user of the system should now be looking at a tiled map which can now be customised before publishing.

4.4.2 Data Store Mining

After establishing a server work flow to map data automatically for a single map, the next step is to apply the work flow to all the data in a data store. The process relies on the fact that a catalogue for a data store will either be available to download directly, or will be machine-readable in some form, whether that is a semantic representation, or even web-scraping. Once a catalogue of all the information contained in the data store has been created, columns are labelled as: IGNORE, TEXT, LINK, TITLE, DESCRIPTION, TAGS, UNIQUEKEY. The 'LINK' field is used to acquire the data from the data store and stage it locally, from where the mining process can attempt to identify the data and make a map from it.

The class diagram in figure 4.35 shows the data store miner base class with specialised sub-classes relating to the London data store, Government data store and NOMIS 2011 Census release. The pattern used is to minimise the amount of code in the specialised classes, which are simple wrappers for accessing data using a data stores catalogue, semantically describing the fields in the catalogue file and, finally, the mechanics of downloading and staging the data locally for processing[19]. The map making process

[19]While the downloading and staging of data might seem simple, the NOMIS Census release contains data for multiple geogra-

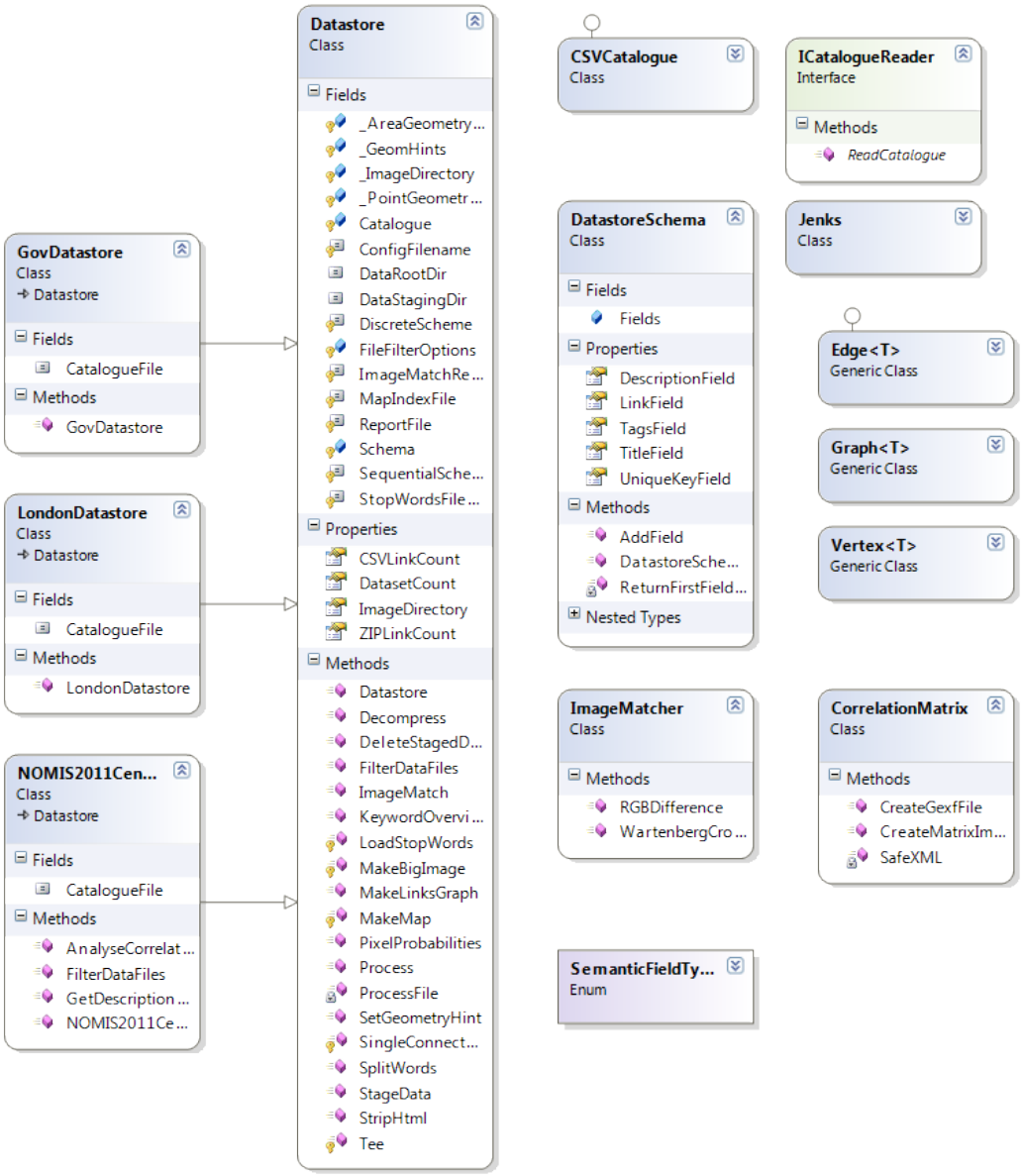


Figure 4.35: Class diagram for the Data Store Miner.

is then just a call to the MapTube library with a reference to the staged data. Additionally, though, the data store miner generates results and statistics files which are useful for the post-mining analysis stages.

Once all the data that can be identified has been processed, the potential is there for data mining algorithms to find any unusual or interesting features. When this was attempted on the NOMIS Census 2012 release, the research aspect to this problem became apparent. Firstly, there are data columns which are repeated and many others which are derived columns, and so correlate very strongly. Also, any data relating to country of origin, language or religion also correlates highly, but for reasons which are

phies in the same zip file, which needs to be decompressed and separated into OA, LSOA, MSOA and LA geography files.

trivial. The key is to find data which correlates highly in the data domain, but which is dis-similar in the semantic domain, or vice-versa. Using the data store miner outlined previously, datasets can be classified on the basis of how similar the data is and also how similar the description is using keywords, tags and plain text descriptions with natural language analysis. Network graphs of how highly connected maps are in each domain can be produced and analysed, but for any significant amount of data, this can be a very computationally intensive process. Comparing every combination of dataset is an $\mathcal{O}(\frac{N^2}{2})$ process while each comparison could potentially involve $288,000$ tests for Output Area level data or $7,200$ tests for MSOA level data. With a spatial cross correlation, this then becomes $288,000^2$ or $7,200^2$ operations for all $\frac{N^2}{2}$ datasets, which is computationally challenging. The NOMIS data contained 2536 identifiable datasets at OA and MSOA level, resulting in $3,215,648 \times 288,000^2 = 266,718,707,712,000,000$ operations while an Intel Core i7-3930K @3.20GHz is quoted at 153.6 GFLOPS[20].

Finally, while data stores are one source of data, ultimately the aim has to be to couple this type of automated system with an Internet search engine and allow it to find its own data on the Internet. The 'MapTube' website uses RSS feeds from various news agencies to build a 'topicality' index for all the map data on the site. This is similar to the referrer systems used by e-commerce websites, for example, "customers who viewed this product were also interested in ...". In this situation, though, any keywords with a high topicality but which do not link to any maps are potential candidates for gaps in our knowledge, which is an opportunity for automatic generation.

[20]Source: Intel specifications, *http://www.intel.com/support/processors/sb/CS-017346.htm*.

4.5 Real-time and Programmable Maps

Real-time data, typified by frames of weather radar animations or positions of tube trains on a network, require a work flow from data acquisition through to the final visualisation. It is this programmable element that distinguishes this section from the previous sections which dealt only with visualisation. Also included with real-time data is the ability to run models on the data and try "what if" scenarios with a view to prediction, both long-term and "now casting" for the short-term. Due to the quantity of real-time information that can be produced by automated sensors, real-time data also has to deal with information management and knowledge directed visualisation, because, in the vast quantity of information available, only a small portion of it is likely to be interesting. The programmable element comes from the fact that the map is now built by a piece of dynamic code, rather than a reference to a dataset. Programmable here can also imply explorable data as there is now direct access to the data and the code controls how it is visualised.

Taking the weather radar example, in the UK this data is available from the Meteorological Office, requiring a capabilities request to ascertain the available observation times, followed by requests for the actual data:

Capabilities Request	*http://datapoint.metoffice.gov.uk/public/data/layer/wxfcs/all/xml/capabilities?key=⟨mykey⟩*
PNG Overlay Request for Radar	*http://datapoint.metoffice.gov.uk/public/data/layer/wxfcs/Precipitation_Rate/png?RUN=2013-09-25T09:00:00Z&FORECAST=0&key=⟨mykey⟩*

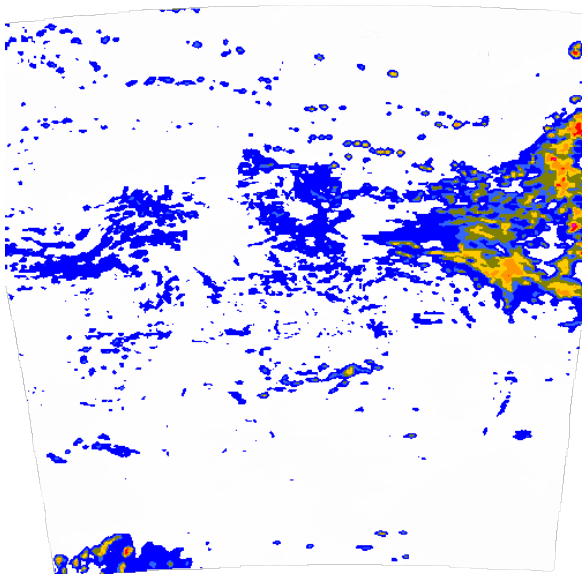


Figure 4.36: Radar image returned from Met Office Datapoint API.

In this case, the returned data is a single PNG image overlay, but recently a WMTS tile server has been added which allows Google Maps and OpenLayers compatible tiles to be requested for use in web-based tiled maps. As with all these types of open data systems, there are "fair usage" policies, so the example in figure 4.32 showing 10,000 requests per minute would not qualify. This identifies another architectural requirement, namely a cache system to de-couple the load presented to the 3rd party system supplying the data and our own servers which take the full load. There are numerous commercial and open-source solutions to this, for example, Microsoft's Application Request Routing (ARR) or HAProxy (*www.haproxy.org*), but this is also something that the MapTubeD system implements with its own tiles. A simple addition would be to add a new data source, specifying a 3rd party tile server as a pass-through. This method would allow limits to be defined for each data source to ensure that the fair use policies of each one were never exceeded. Applying this to the second example of real-time transport systems, data for all London Underground tube trains can only be queried once every three minutes[21]. In this instance the raw data requires significant pre-processing to calculate vehicle positions, with real-time processing of Network Rail trains, London Underground Tubes and TfL buses and River Services the subject of a two month project called "Adaptive Networks for complex Transport Systems" (ANTS) in May 2012[22]. The model used here is to create a library to acquire and process the data, with common code for both off-line analysis of long-term data and real-time instantaneous views. A REST API then uses this library to provide the data to any internal systems that request it using a 3-tier structure and relational database allowing access to the last 24 hours' worth of data. Figure 4.37 shows the flow of data from the API or stream though to the derived vehicle positions. This includes making comparisons to baseline running data from the archive, which is used to identify any late running services. Due to the data coming from three different sources and in three different formats, this is as much an exercise in data fusion as it is real-time analytics. More information on the 'ANTS' system and real-time London transport systems is published in [Che+14].

While the 'ANTS' library is used for processing the data, an additional web server component is required to enable other systems to access the data. The class diagram in

[21] The three minute cycle is accurate as of August 2014.

[22] The ANTS project was a 2 month project in May 2012 which was funded by Future ICT and resulted in a number of real-time transport data publications including: [Bat+13] and [Che+14].

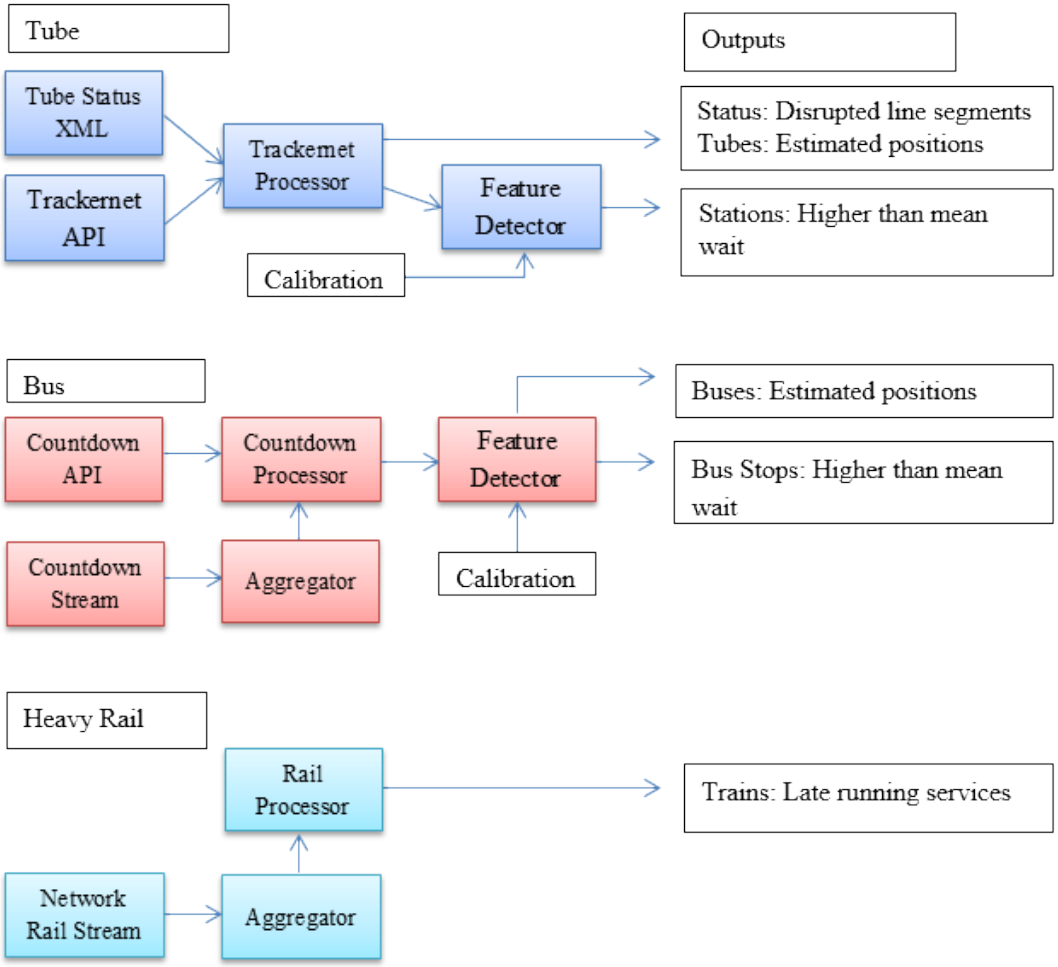


Figure 4.37: System diagram of data flow in the ANTS system.

figure 4.38 shows the implementation of the DataAPI. The pattern employed is to create an 'ICityDataSource' interface which all data sources implement to acquire data. A further 'ICityChildDataSource' interface is required for data sources which are derived from other data sources. For example, 'Trackernet' is the name of the TfL system providing open data on the status of the London Underground. The 'TrackernetDataSource' acquires this data via an HTTP request and derives positions of all the tube trains. The 'TubeNumbers' class is a child data source as it uses the data from the 'TrackernetDataSource' to count the number of tubes running on each line at that instant in time. The 'CountdownDataSource' is the TfL system for bus tracking, which in this case is a stream API, with 'BusNumbers' the total number of buses running in London. 'NetworkRailDataSource' and 'NetworkRailNumbers' provides a similar breakdown of Network Rail trains by train operating company (TOC).

Data for the last 24 hours is stored in a Microsoft SQL Server 2012 database, using

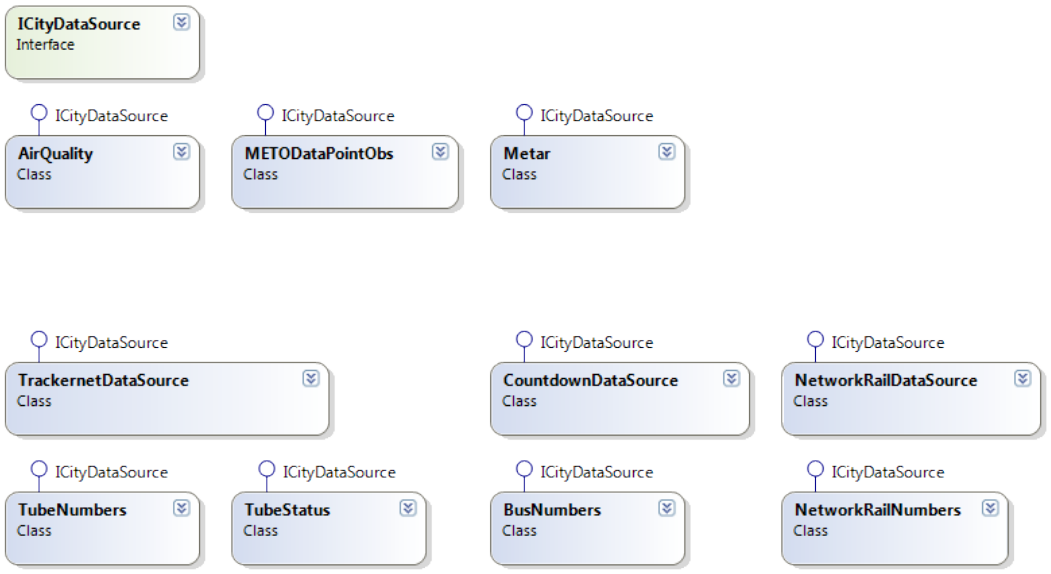


Figure 4.38: City Data source Class Diagram.

LINQ to SQL to generate the data classes automatically. This system then becomes another architectural element for providing real-time data required by online visualisation systems, but only after an API has been added to provide a means for accessing the data. This was modelled on the 'restSQL'[23] idea of providing limited querying ability to SQL databases via HTTP REST. The query either takes the form of a request for filtered data, for example, the numbers of tubes on the London Underground over the last 24 hours (select query), or a query for a raw data file which might contain the latest tube positions for all lines (file query). All data downloaded from data APIs is stored on the server as files which are timestamped for easy access. The syntax of the two types of query is shown in table 4.2. Finally, the system archives all raw data downloaded which it stores on a local storage array for future data mining or any other long-term analytics.

Using the architecture just described as a new data source, it now becomes possible to create real-time visualisations based on London's transport system at the current instant in time. Due to the complexity and nature of the data, the amount of useful information that can be extracted from a simple plot of vehicle positions is very limited. One example was during the tube strike of 29th April 2014 when Victoria Line trains were seen to be running as far south as Brixton when the official line from TfL was that they were turning around at Victoria. This type of obvious data is easy to extract, but,

[23] REST SQL website: *http://restsql.org*.

Table 4.2: Real-time Transport API Web Services REST Syntax.

Real-time Transport Data API Syntax

Select Query Syntax:	
Usage	REST URI
select 24 hours from {table} as json	*api.svc/s/{table}/json*
select 24 hours from {table} as xml	*api.svc/s/{table}/xml*
filter based on table column matching value	*api.svc/s/{table}/json?column=value*
limit returned results to {int} rows	*_limit={int}*

Examples:
1. Select the 12 most recent tube number counts for all lines:
   api.svc/s/trackernet/json?_limit=12
2. Select the 12 most recent network rail number counts for train operating company (toc_id) 86:
   api.svc/s/networkrail/json?toc_id=86&_limit=12

File Request Syntax:	
Usage	REST URI
return specific archive file	*api.svc/f/{dir}?pattern={wildcard}*
file pattern wild card to match any combination of characters	*

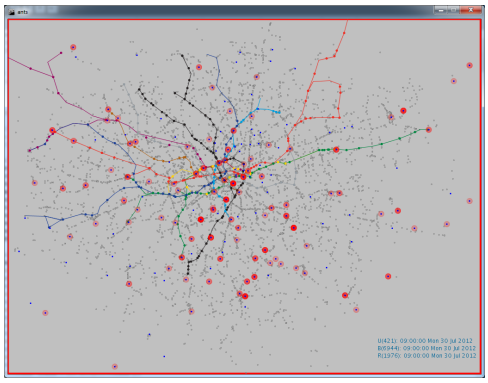
Examples:
1. Download latest national rail file containing all train positions:
   api.svc/f/nationalrail?pattern=nationalrail_*.csv
2. Download a buses (countdown) file for the latest available time between 09:00 and 09:59 on 11th September 2014:
   api.svc/f/countdown?pattern=countdown_20140911_09.csv*

for anything more complicated, a mathematical comparison between normal running and the current situation is required. This is where knowledge directed visualisation becomes essential.

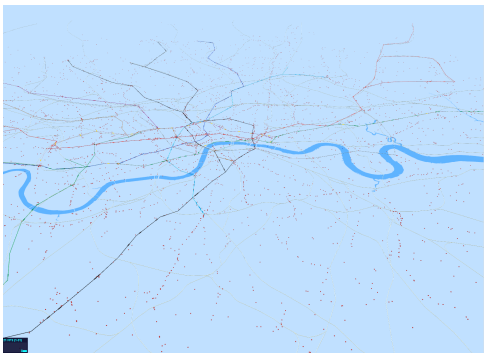
Visualisation of vehicle positions with three minute resolution data also poses some problems due to the periods between data updates. Having positioned a tube on the network, no new data is available for another three minutes, so any real-time visualisation showing animated tubes is forecasting positions into the future based on the best information available at the time. This uses the reported time to next station to predict the arrival, but it is perfectly possible that the new data will show the vehicle as stationary and the forecast position will be wrong at the next data update. A first attempt at the real-time visualisation problem is available at: *https://github.com/maptube/FortyTwo*. This is a Javascript and WebGL visualisation in 3D showing real-time tubes and buses. The data is extracted from the 'ANTS' server described earlier and contains vehicle positions as latitude and longitude for the last time point.

The visualisations in figure 4.39 fall into one of two programming paradigms, being based on either a key frame animation model, or on agent based modelling. The data contains positions of vehicles at specific points in time (key frames), so the first method

simply animates positions based on linear interpolation. This is fine for archive data, but with real-time data where the future position of a vehicle is not know, the technique is extended to use the vehicle's expected arrival time at its next location. While this method works, its limitation becomes clear as time continues to move forward. When a vehicle arrives at its next location and before the next data update giving a new estimated time of arrival, the simulation has to work out the vehicle's route and estimate a new arrival time based on the data available. The key word here is 'simulation', as a real-time visualisation is a simulation of the network over time, based on behaviours built into the agents. In this case, a network graph of the London Underground network with times between stations recorded from the official timetables or archive running data which are used to make arrival time estimates.

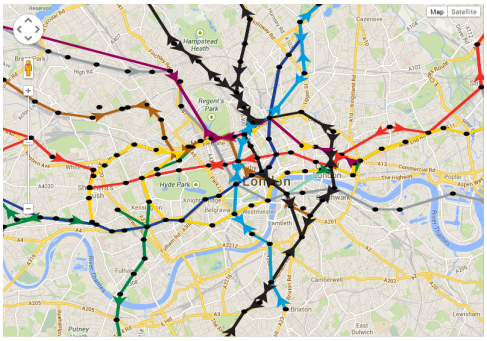


(a) Tubes, buses and trains using Processing.

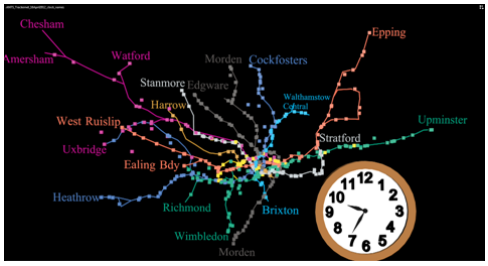


(b) Tubes and buses in Chrome and WebGL.

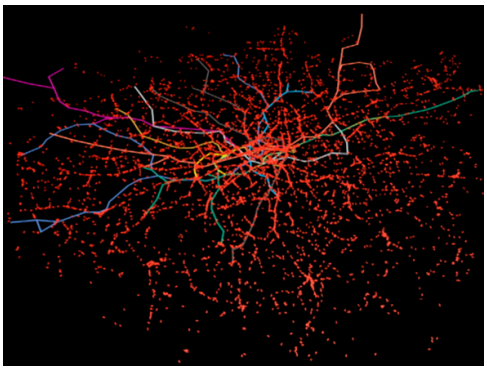
(c) Tubes in Chrome and AgentScript.



(d) Tube strike day one.



(e) Tube animation using Autodesk 3D Studio Max.



(f) Bus animation using Autodesk 3D Studio Max.

Figure 4.39: Six visualisations of London's real-time transport data.

4.6 Conclusions

Having outlined some existing real-time geospatial systems, the need for a reusable library of core functions is clear, along with the need for interoperability between systems. While many of the functions are not new, the way that these fundamental building blocks are being used suggests a new approach. The tiled mapping systems previously described demonstrate the rendering layer of a GIS system with scalability to millions of users. While systems like Google Maps, Bing Maps and OpenLayers have made significant advances in basic data visualisation, a case can be made for a specialised data visualisation system rather than tile renderers which are based on cartographic rendering adapted for data in the form of choropleth maps. Some of the client-side data visualisation libraries, for example D3, have moved towards geographic visualisation with dynamic maps showing real-time flows of wind patterns[24] or voting behaviour[25]. These are a combination of bespoke development and existing libraries, but show how new data visualisation techniques can be applied to geographic data using modern computer hardware. As computers have become more powerful, the quantity of data which a desktop machine can handle has increased. To quote from Hennessy and Patterson:

> "For many applications, the highest performance microprocessors of today outperform the supercomputer of less than 10 years ago."
>
> (Hennessy and Patterson, [HP11, pp2])

Add to this the GPU performance statistics in the chapter on parallelism and GPU architectures in [HP11, Ch4] and the links between heterogeneous computing and MapReduce from [Gas12] and a scalable geospatial computing architecture begins to form. With the aim of building a realistic real-time city model which researchers can use to experiment on, where data is ingested from numerous locations in a variety for formats, or for mining data stores for information, work flows of recognised geospatial operations are required. While organisations like the OGC go a long way towards formalising standards, this does not go far enough in the case of the actual operations themselves. Looking at the data mining tool, WEKA, and also the MOA tool for 'Big' data mining, these are Java libraries with well documented methods explaining the mathematics behind the tools and how to use them programmatically. This is more like the GRASS geospatial library which can be used to build a work flow using command

[24] Global wind animation: *http://earth.nullschool.net/*.
[25] U.S. voters: *http://www.nytimes.com/interactive/2012/11/11/sunday-review/counties-moving.html?_r=0*.

line tools, or the Java based Geotools library which is the foundation of many higher level GIS systems. The 'CyberGIS' approach to spatial analysis advocated by Anselin, Goodchild and Wang in PySAL [Ans12] is a heterogeneous cloud computing solution. While the lines dividing the different architectural approaches are blurred, the need for inter-operability between systems could not be more clear. In essence, the work flow itself is what is important, not how it is being achieved.

To conclude, the city data model relies on data mining, data store mining, real-time analytics, agent behaviours and geospatial visualisation. The automatic map generation and data mining components enable the use of entire data stores as sources of information, which marks a change from analysing single geospatial datasets to analysing connected sets of inter-related data. The real-time aspect increases the quantity of data available and the speed at which it must be processed to remain current, all of which points to parallel computing architectures which are able to process, mine and correlate data in time-scales which are useful to researchers. Data-driven simulation is the other key aspect of this system, with the long-term archives of data currently being produced by real-time city systems used to derive statistically accurate models of behaviour which then allow experimentation on a real city.

The aim of the next two chapters is to put all these elements together in a fully functioning system. The MapTube website and NOMIS Census 2011 data store are central to the argument for being able to run analytics work flows against stores of connected data and so enable mining of data stores. Automatic map creation is a tool for increasing the amount of geospatial data available, but with the added benefit that it can also make analysis easier if the identification, cleaning and preparation of data are part of the automatic work flows. Data in data stores can be sorted into groups by similarity, giving rise to different methods which can be used to search a data store for data. This is perhaps the biggest potential benefit, being able to turn a data dump into usable knowledge by identifying links between datasets. When real-time data is looked at in the later chapter, these techniques are turned to learning from streams of data in order to understand real-time city systems like the tube, bus and rail networks. All of the data being handled has a spatial context, so the basis of the tiled mapping presented in this chapter is expanded into a provider of data for both 2D and 3D systems, bringing all of the techniques together. Real-time data exists on the Internet because it can be updated and disseminated to a wide audience in real-time, so lower end 2D mapping

systems provide access to this with simple visualisation functions, while higher end systems consume their output to provide more complex analytics. A common thread ties together the acquisition, storage, identification, analysis and visualisation of data in both 2D and 3D systems, which is explored further in the next two chapters.

Chapter 5

Dynamic Visualisation

This chapter looks at the requirements covered in the previous chapters and builds on previous work to build a next generation geographic analysis and visualisation framework. The base algorithms identified in the requirements phase are built into a distributed system capable of handling today's bigger and faster work flows in a way that scales linearly with size. Work flows for handling 3D and time series data from multiple sources in a coherent way will be explored. The chapter starts by introducing a basic geospatial data processing framework which is then extended to fulfil the requirements of three questions which increased computing power and storage now allow us to ask. The following quote from the 1987 book "2061" by Arthur C. Clarke shows how advances in computing power have exceeded expectations, despite the Moore's Law predictions:

> "I too take leave of all I ever had..." From what depths of memory had *that* line come swimming up to the surface? Heywood Floyd closed his eyes, and tried to focus on the past. It was certainly a poem - and he had hardly read a line of poetry since leaving college. And little enough then, except a short English Appreciation Seminar.With no further clues, it might take the station computer quite a while - perhaps as much as ten minutes - to locate the line in the whole body of English literature.
>
> (2061, written by Arthur C. Clarke, first published 1987)

A Google search for the phrase, "I too take leave of all I ever had ", returns with 223 million results in 0.46 seconds. Apparently the poem is "Farewell" by Robert Nichols, written in 1915. Apart from showing the power of modern cloud computing, the other reason for including this example is that it is also possible to search for 'E02000001' and information about the area from the ONS and Ordnance Survey websites. The

ONS linked data viewer is at: *http://statistics.data.gov.uk/doc/statistical-geography/E02000001*, which contains a SPARQL endpoint and also a link to the GeoJSON boundary file. In fact, just searching with a general Internet search engine can now reveal a wealth of geographic information, including the raw datasets used to make the maps on CASA's MapTube[1] website (figure 5.1), which is explored in more detail later in the chapter.

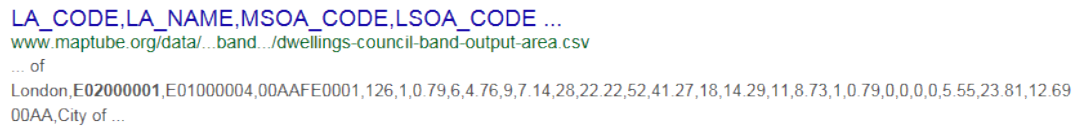


Figure 5.1: A segment of the Google search results for 'E02000001' returning a line from a datafile on CASA's MapTube website used to make a map of housing stock. See: http://www.maptube.org/map.aspx?mapid=868 for one of a number of maps showing the breakdown of housing stock by council tax band.

This opens up the possibility of algorithms which can find geographic data stored on the Internet using semantic information, much like a search engine searches html pages. With information contained in the file semantically defined, it is then a simple matter to link the aspatial data with the correct boundary file data to give it a spatial context and map the data.

To sum up, the quantity of data that can now be handled has increased, while new techniques enable automatic analysis and processing of data which is starting to be linked together, allowing new discoveries to be made.

This is an era of asking interesting questions.

5.1 Example 1: Data Exploration and Web GIS

The first example covers exploratory multivariate data visualisation, focusing on work flows for making maps and performing more complex spatial analysis on the data. While this is the primary function of a desktop GIS, data is now increasingly web based, so the requirement for a 'Web GIS' which goes beyond Google's Fusion Tables is starting to be realised. This, coupled with the uptake of HTML 5, means that the technologies originally intended for browser based 3D games can be harnessed for geospatial analysis. The direct access to the graphics hardware provided by the browser through 'WebGL' now enables raw data to be visualised directly on a web page. This direct access to the data, which is now local, rather than on a remote server, enables

[1] MapTube website: *http://www.maptube.org*.

higher levels of interactivity with lower latency.

The problem to be solved in this section is to build a web GIS capable of visualising complex multivariate data while allowing the user to interact with the data, perform analysis and explore it. Discovery emerges from how new data can be related to existing data and how existing data can be explored in new ways. The approach taken to achieve this is to employ heterogeneous computing and a web service framework providing the functions which cannot be executed on the browser.

As an example of a rich source of data, the UK Meteorological Office's DataPoint website is used and the example is built around weather visualisation. This is a source of data that is global, real-time and multivariate, containing both coverage data (radar and satellite) and point data (synoptic). Additional datasets from the 2011 Census and Internet data stores are then added to cover the requirement for general desktop GIS in the Internet era.

5.1.1 Storage and Retrieval

Cloud storage and Internet data stores are both sources of data, but this needs to include APIs for extracting real-time data, which brings in the concept of a data source as the final end of a work flow[2]. Microsoft OneDrive, Dropbox, Amazon Web Services and Google all provide cloud storage for data which is identified by a unique URI. This is all that is needed, as datasets can be uniquely identified and accessed as required, while the problems of pulling large amounts of data across the Internet can be delegated to warehouse computing and other forms of efficient local buffering. The key point remains that large datasets need to be held close to where they are being used for efficiency reasons. Referencing data on the web using URIs has advantages for web based mapping and sharing of data in the publishing scenario.

5.1.2 Points, Lines, Polygons

The layered diagram in figure 5.2 shows how the different technologies are built on one another to give GIS capabilities to an HTML 5 browser via WebGL. A custom overlay is created in Google Maps which contains the WebGL view. Then a 3D scene is created with the ground on the XY plane and a fixed camera at some height in the Z axis, looking directly down towards the ground plane. Figure 5.3 shows this set up, which is essential to the following discussion. While figure 5.2 shows the OGC's "Simple

[2]Complex data sources like the positions of London Underground trains do not use data directly from the API, but require processing to turn the data returned from the API into a suitable format. The real time element means that this must be a work flow to pull in the latest data.

Features for SQL" standard, along with the "GeoAPI" and "Web Features" standard any consistent representation for the data will suffice. The modular approach chosen here is key to this, allowing any of the modules to be replaced with a different standard when required. As standards are continually changing, any reliance on a specific standard is a bad approach to take, so being standard agnostic where possible is desirable. However, standards allow for inter-operability between users and internal components, so the standards that are exposed to the outside world are important. Having said this, though, the digram in figure 5.2 is intended to show how the realisation of an actual system was constructed, so the standards used internally will never be exposed and can be chosen on the basis of system efficiency.

All geographic data is rendered on the $Z = 0$ plane, with the camera positioned at $Z = 400$ and using an orthographic projection. The near and far clip planes are at $Z = 1$ and $Z = 1000$ respectively. Equation 5.1 shows the orthographic projection formula, consisting of scaling and translation. The symbols 'L', 'R', 'T' and 'B' are the left, right, top and bottom coordinates of the viewport. The distance to the near clip plane is given by 'N', while the far clip plane is 'F'. The projection matrix, P, is calculated from the scaling and translation matrices as follows:

$$P = \begin{bmatrix} \frac{2}{R-L} & 0 & 0 & \frac{-R+L}{R-L} \\ 0 & \frac{2}{T-B} & 0 & \frac{-T+B}{T-B} \\ 0 & 0 & \frac{-2}{F-N} & \frac{F+N}{F-N} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{2}{R-L} & 0 & 0 & 0 \\ 0 & \frac{2}{T-B} & 0 & 0 \\ 0 & 0 & \frac{2}{F-N} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & \frac{-L+R}{2} \\ 0 & 1 & 0 & \frac{-T+B}{2} \\ 0 & 0 & -1 & \frac{F+N}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

Geographic data is initially specified in WGS84 and then projected into Spherical Mercator coordinates which is the coordinate system of the Google Map layer beneath the data.

$$x = R\lambda, \quad y = \frac{R}{2} \ln \left[\frac{1 + \sin \phi}{1 - \sin \phi} \right] \tag{5.2}$$

This is extractable from the source code of the 'map.getProjection().fromLatLng ToPoint()' function in the Google 'maps.js' library. The example in equation 5.2 was taken from version 20.2 of the V3 API and is liable to implementation changes. What has been omitted here for clarity is that the Mercator coordinates are then scaled to

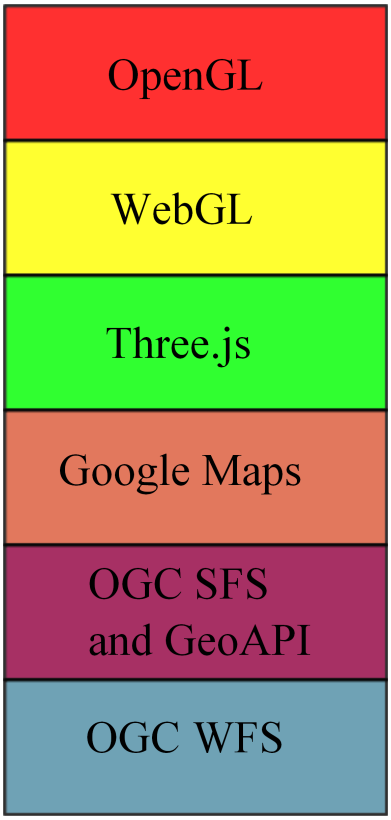


Figure 5.2: MapTube WebGL System Diagram.

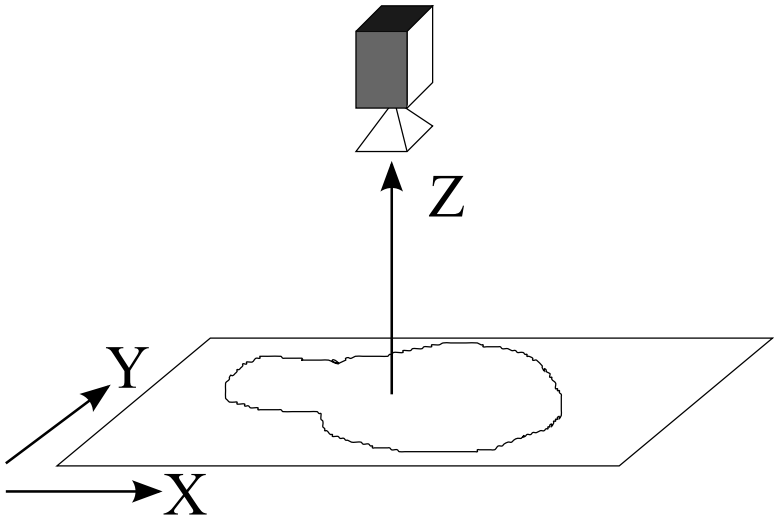


Figure 5.3: MapTube WebGL Axis System.

a 256x256 unit box to simplify the conversion to pixel coordinates at arbitrary zoom levels. The tiles are also 256 pixel square boxes, starting with the whole world fitting the single top level tile, then the size doubling at each successive zoom level. This means that the coordinates are multiplied by 2^z at each zoom level, z, in order to map them onto the pixel canvas.

The geographic bounds of the data form a box, which is projected onto the Google Map and then used to create the rendering canvas (frame buffer) for the geographic data to be drawn on. This defines the size in pixels of the drawing surface. As the map is zoomed in, this canvas doubles in size in both the X and Y dimensions. The main issue with this system is that it also requires the inclusion of a clip rectangle to limit the rendering canvas to the size of the screen. Otherwise, the x and y frame buffer dimensions for the whole world are defined by $x = 2^z$ for zoom, z, between 0 and 20. This can obviously become very large, so a clip rectangle which fits the screen viewport needs to be included, along with the required offsets to ensure the data is rendered in the correct place on the map.

The diagram in figure 5.4 shows the calculations required to transform the original data into the 3D world space and render it in the correct position on the Google Map *'div'* container for the custom overlay. The first box, labelled 1, shows data in WGS84 geodetic coordinates projected into the world space of box 2. Taking the bounding box of $(55.8, 2.0)$, $(49.8, -6.5)$ used in the example, which encloses England and Wales, the coordinates projected into the Mercator world space occupy $X = \{0..256\}$, $Y = \{0..256\}$. The camera is always placed at the centre of the data bounds, which in this case equates to $(126.4, 83.5)$. This projection is performed using the Google maps function, *'map.getProjection().fromLatLngToPoint()'*. In order to calculate the position that the overlay's *'div'* container occupies on the map, the original data bounding box from (1) is projected onto the current visible map overlay using the Google Maps function, *'overlay.fromLatLngToDivPixel()'*. This is how the scaling is handled between different zoom levels, as the camera scale is calculated from the ratio of the ordinates of the bounding boxes calculated in (2) and (1). This takes care of the camera position and scale, but, due to the map canvas doubling in size at each zoom level, clipping of the frame buffer to the visible viewport is required. This is accomplished using the *'overlay.fromLatLngToDivPixel()'* function again, but this time on the geographic bounds of the map. Step 4 shows this process, which results in the data bounds enve-

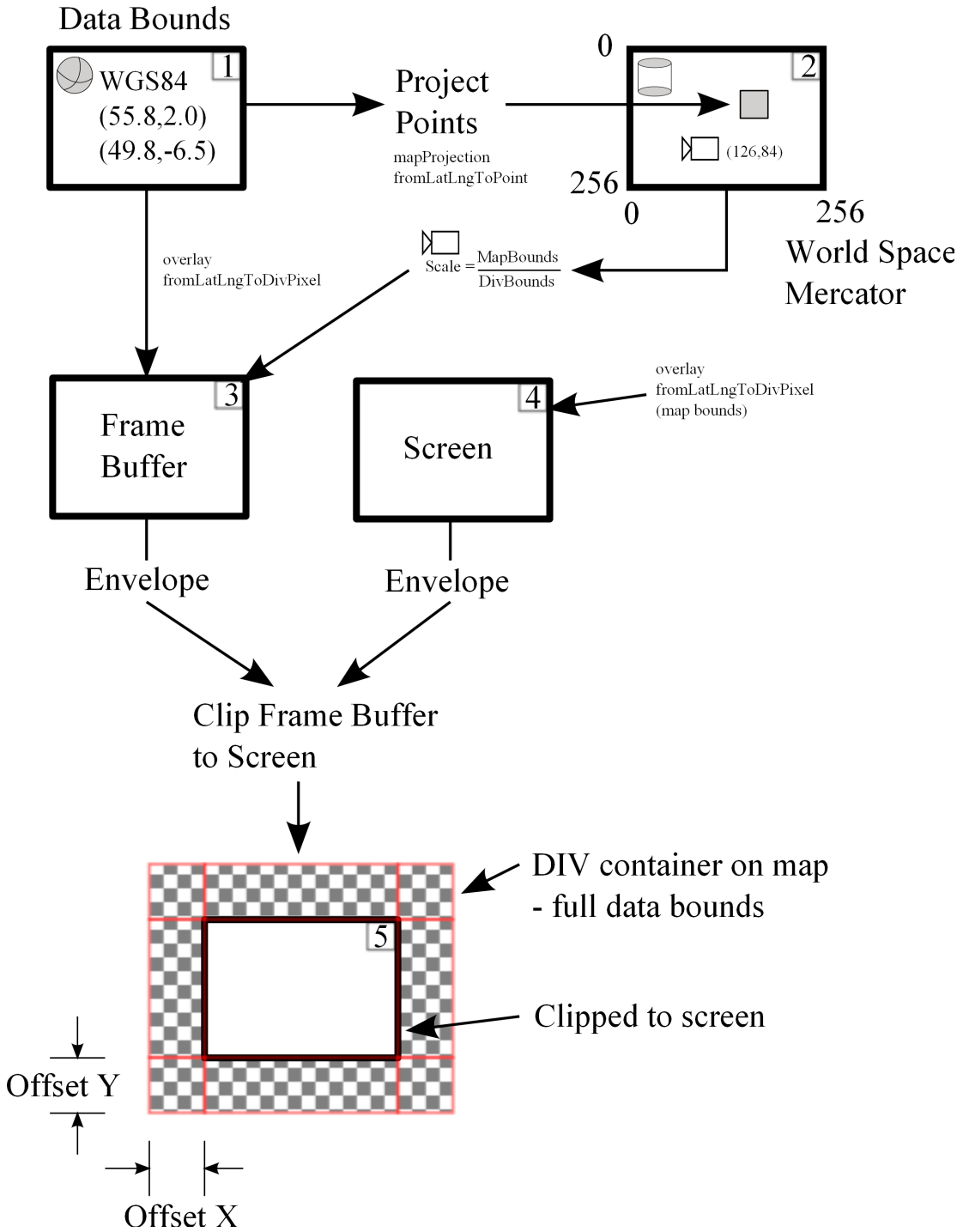


Figure 5.4: MapTube WebGL Projection Calculations.

lope from (3) being clipped by the screen extents envelope from (4). The final frame buffer size is shown in (5), with the chessboard pattern intended to represent the parts of the data frame clipped by the visible viewport. The offsets here are required for rendering, otherwise the data would be translated on the map due to the viewport clipping. The camera projection matrix is recalculated using the revised (offset X, offset Y) and (width, height) ordinates, after which the scene can be rendered in the correct position on the map's frame buffer which fills the Google Map overlay *'div'*.

For similar reasons to the 'need for a renderer' covered with the 3D globe system later in section 6.3.1, the 'THREE.js' library is used here to manage the WebGL context and provide scene and object creation functions which the geographic data layers use to draw their data on the map. Objects in the scene are created using Google's projection of the original WGS84 coordinates into the 'Google Mercator' projection where the world is normalised to fit in a box measuring 256x256 units. Figure 5.5 shows an example of this system.

Comparisons between this system and the AgentScript system from section 3.9 are inevitable as the basic code pattern is very similar. The difference is in the use of the faster 3D graphics hardware for rendering the maps and some additional projection calculations in the frame buffer clipping. While this was something that was explored in section 4.2 on the background to tiled vector maps, the Cartagen and MapBox systems mentioned did not use a Google Maps base layer and were intended for use with cartography. The focus here is to provide visualisation of intangible data rather than tangible cartography, which is aided by the ability to handle more data, animation and a pre-built base map in the form of Google Maps. Recent advances in MapBox GL JS since October 2015 are significant here, providing an open source project which can be used for vector maps on the browser.

While this is a useful test, some questions still remain about how 2D and 3D GIS can be integrated. The system diagram in figure 5.2 shows two OGC blocks, where the 'Web Feature Service' is used to query data on a server and the 'Simple Features for SQL' and 'GeoAPI' block handles the scene creation from the data. Unfortunately, these standards are not quite usable yet for 3D GIS. Points and lines can be easily represented, but polygons require a triangulation step which is computationally expensive. While the client side library is able to handle this, it makes more sense to perform this step on the server and pass triangulated data directly to the client. Geospatial standards

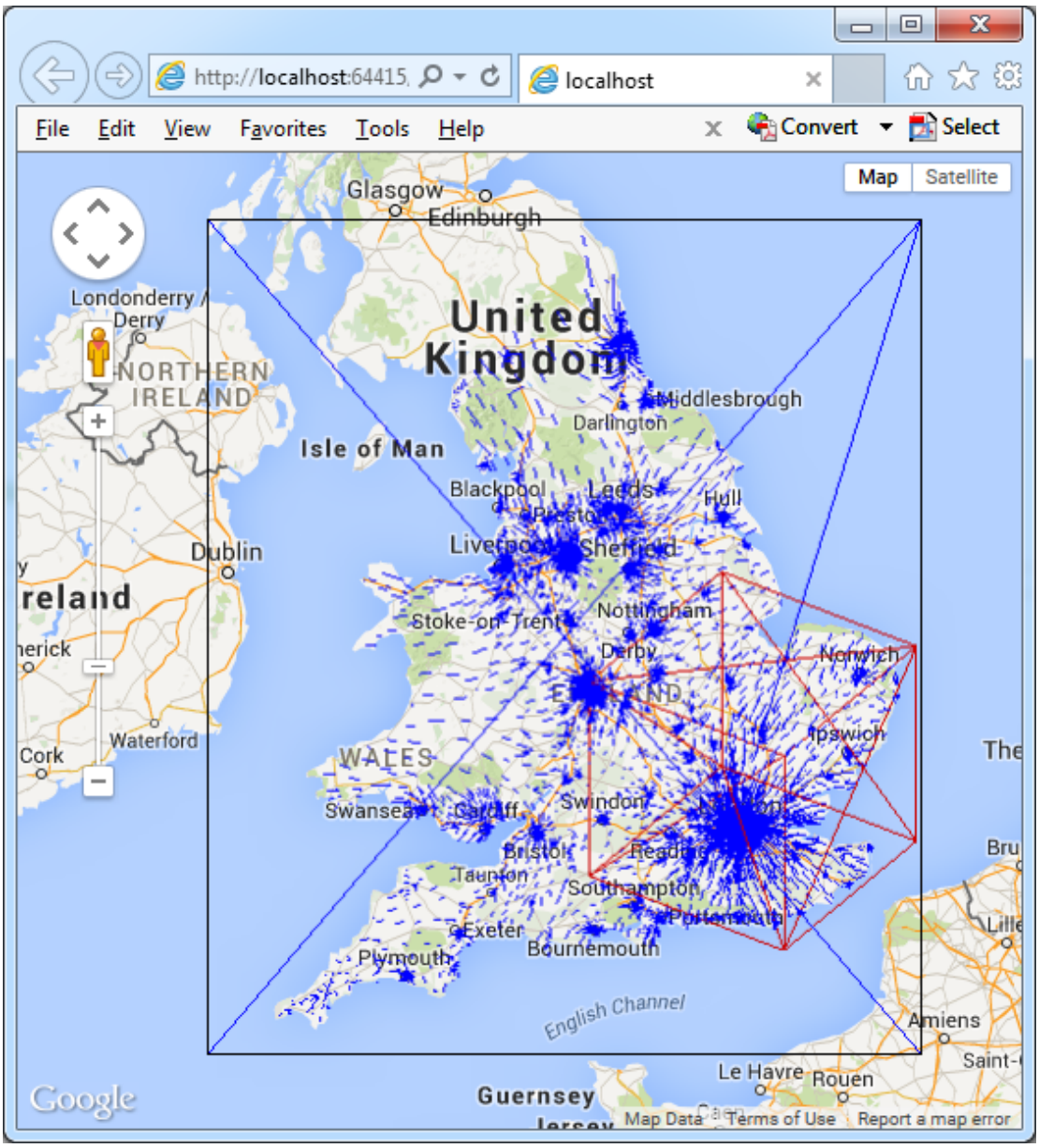
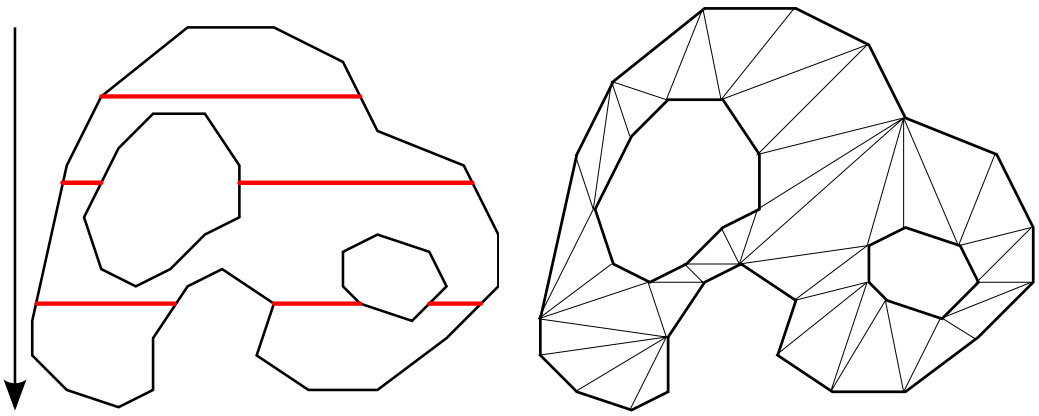


Figure 5.5: MapTube WebGL Layer inserted into Google Maps. The data shows a test of commuter flow lines from the centroid of every one of the 7,201 MSOA areas in England and Wales in the direction of the average flow (blue arrows). The geographic bounds of the data are marked with the black box containing a blue cross. The red wire frame 3D cube was added to show that this is a 'live' scene graph, with the cube slowly rotating over the map. The potential exists here for animated data on web based maps.

for this type of operation have yet to become fully accepted, so an interim solution is the most likely solution. In addition to this, the 'Web Feature Service' is intended to be used with GML data, although there is a provision in the standard for advertising alternative data formats. GML is a form of XML mark-up for geographic modelling and is not a natural candidate for use with web browsers, which are extremely limited in the data formats that they can handle. An obvious choice is to create a WFS service that returns JSON[3] which is more efficiently handled by the browser. Then the problem of the format of the data needs to be tackled, as a form of GeoJSON containing triangulated data does not exist and any visual representation of the polygon's outline would still require the outer boundary as a linestring. At present, Google's implementation of the OGC features specification uses points, linestrings, polygons and their respective 'multi' containers. Polygons are specified using one outer boundary and zero or more holes, which is the standard practice. The 'google.data.feature' object contains properties and geometry data, which is used to render styled features on the map, but uses a 2D canvas to draw the data. In order to draw the shape, this uses a technique called a 'Path Geometry', which is based on drawing scan lines from top to bottom with horizontal spans between two outer boundary points split by intersections with the inner boundaries. The two diagrams in figure 5.6 show the difference between the scan line approach and full triangulation.



(a) Scan line algorithm, with three scan lines taken through the object shown in red.

(b) Triangulation algorithm, showing one of a number of valid triangulations.

Figure 5.6: A geographic object containing two holes, showing the 'scan line' method for rendering a 2D shape, compared to the triangulation required for 3D rendering.

[3]Using JSONP avoids any cross origin security problems.

5.1.3 Exploratory Data Analysis: Weather Data

Using the techniques explored so far, this section builds a weather data viewer for the web using OGC compliant web services[4]. In addition to using real-time data, archive data can also be explored, for example the storm of the 27th and 28th October 2013. The 'HadCRUT 4' dataset is also available for download and contains climate information for the whole world spanning 1850 to the present day. Global Telecommunications System (GTS) data is also freely available from some organisations that have access to a GTS feed and can be decoded using the following software: *https://github.com/maptube/WxDecoder*. Linking back to the flooding example in section 3.2, in March 2015 the Environment Agency released a new real-time API for river level data and flood warnings, so the inclusion of hydrological information is now a possibility. All these sources of data contain multiple variables for points in space and time, fulfilling the criteria of multivariate exploratory data analysis.

Table 5.1 lists the map based products available on the UK Meteorological Office's DataPoint site for real-time weather data. Table 5.2 lists the forecasts that are available, excluding text only forecasts and other non-mappable products like fax charts. One of the problems with integrating weather data into modern geographic display software is that these weather forecasting data sources have existed since the 1940s and are designed to remain accessible in remote parts of the world where computing power is scarce. Some of the primary resources for weather forecasting, for example ASXX surface pressure observation charts and FSXX surface pressure forecast charts, are in raster formats originally designed for dissemination by fax machine.

Table 5.1: Met Office DataPoint Observations.

Name	Availability	Format	Spatial Extents	Projections
Surface Pressure Chart	every 6 hours	png	Northern Polar	Polar Stereographic
Rainfall Radar (Single Tile)	every 15 mins at T+15	png	(48.0,-12), (61.0,5.0) WGS84	Mercator Image
Rainfall Radar (WMTS)		WMTS	EPSG:4258, ETRS89, EPSG:27700, EPSG:29903, EPSG:2157	
Satellite (Visible)	every 3 hours at T+15	png	(48.0,-12), (61.0, 5) WGS84	Mercator Image
Satellite (IR)	every 3 hours at T+15	png	(48.0,-12), (61.0, 5) WGS84	Mercator Image
Radar and Sat Composites	every 30 mins	png	(48.0,-12), (61.0,5.0) WGS84	Mercator Image
Temperature Colour fill	every 6 hours	png	(48.0,-12), (61.0,5.0) WGS84	Mercator Image
Hourly Observations	Hourly at T+10	XML	UK (WMO 03)	WGS84
Marine Observations	Hourly at T+10	XML	UK (WMO 03)	WGS84
Monthly Climate Averages	Daily table of data	csv	n/a	n/a
Lightning Strikes	every 15 mins at T+15	png	(48.0,-12), (61.0, 5) WGS84	Mercator Image
UK Climate Averages	Daily table of data	csv	n/a	n/a
Historic Station Observations	Table of data	csv	n/a	n/a

[4]UK Government Agencies are directed to follow the European INSPIRE directive for the dissemination of geospatial data. This in turn uses OGC standards.

Table 5.2: Met Office DataPoint Forecasts.

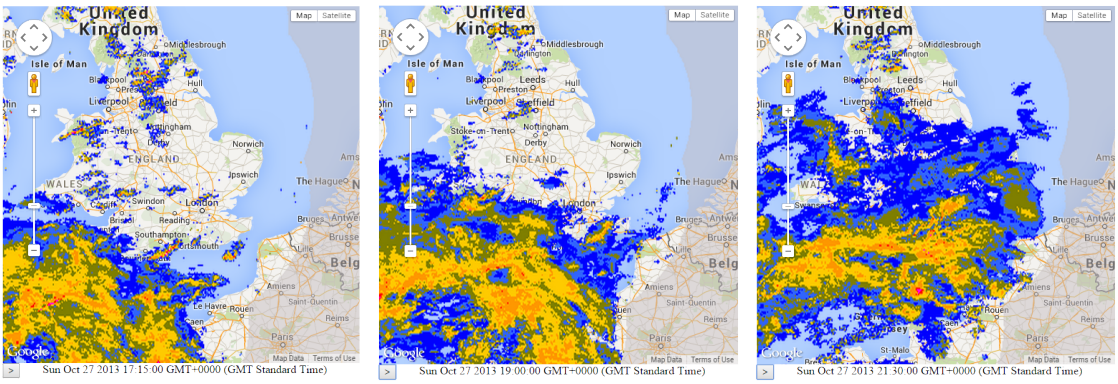
Name	Availability	Format	Spatial Extents	Projections
Precipitation Forecast	every 6 hours	png	(48.0,-12), (61.0,5.0) WGS84	Mercator Image
Surface Pressure	every 6 hours	png	Northern Polar	Polar Stereographic
UK 3 Hourly Forecast	3 hourly at T+30	XML	UK	WGS84
UK Site Specific Forecast	3 hourly	XML	UK sites	WGS84
Forecast Radar	hourly	png, WMTS	(48.0,-12), (61.0,5.0) WGS84	Mercator
Forecast Sites	forecasts for 5000 sites	xml	n/a	WGS84
Total Cloud	every 6 hours	png	(48.0,-12), (61.0,5.0) WGS84	Mercator Image
MSLP Forecasts	T+0 to T+36 (+3) updated every 6 hours	png	(48.0,-12), (61.0, 5) WGS84	Mercator Image
Temperature Forecast	every 6 hours	png	(48.0,-12), (61.0, 5) WGS84	Mercator Image

For the purposes of geospatial visualisation, the data can be categorised into either image or synoptic data. The synoptic data can be handled using the MapTube WebGL system outlined in the previous section 5.1.2, while image, or 'coverage' data from satellite and radar instruments needs to be handled differently. The reason for this example is that it demonstrates custom rendering of data on web-based maps using weather data, which is rich in custom symbolisation, as an example. The wind fleches, cloud type symbols, oktas of sky coverage and past and present weather symbols all require custom rendering code, which is accomplished by drawing onto the frame buffer. Although the synoptic data is point based, while the coverage data is image based, the custom code required to draw the meteorological data on the frame buffer is accomplished by "MapTube.ImageOverlay" class in figure 5.8. Figure 5.5 showed a more complex example using the THREE.js library to show flow lines, which could be adapted for any of the synoptic observation data in tables 5.1 and 5.2. The previous discussion in section 5.1.3 contains references to the system architecture which shows how all of the custom rendering code is reduced to calls of the form, "move x,y", "draw x,y", or "draw text at x,y". The architecture presented in the GitHub code abstracts away the tile rendering complication and makes the rendering of custom data more like drawing to a screen canvas in a desktop application. This is essential to the following discussion in the later chapters on agent based models running on web-based maps, as the requirement here is identical. What follows in this section is a discussion about the problems of the canvas size, which multiplies by four at every subequent zoom level.

The temporal element also adds an additional dimension to both types of data, but, by using the full area 'png' images, which are already projected into the EPSG:900913 coordinate system, the data is representable using a bounding box overlay on top of either a Google Map or using OpenLayers. The documentation for the Google Maps library includes an example of data from the USGS, overlayed on top of a Google

Map using this technique. Internally, the web browser represents the image overlay as an HTML image element, *'<img>'*, contained within the Document Object Model. This raises the same performance issues which were explored in the previous WebGL section, namely the doubling of the image X and Y dimensions each time the user clicks the map's zoom in button. In this situation, the image scaling and clipping is handled natively by the browser and operating system, so this problem has not been experienced to the same degree as the WebGL overlays. However, with a sufficiently large bitmap image, this is a potential problem, so a proposed solution is to use the same WebGL system for the image data by texturing a polygon covering the data's bounding box. This idea raises an interesting possibility, as more detailed rainfall and satellite data is available in the OGC's WMTS format, but not in the web Mercator projection required. The equations used by the virtual globe system detailed in Chapter 4 of [CR11] take the NASA Blue Marble texture for the world in EPSG:4385 and wrap this around the spheroid using the programmable GPU shaders to do the coordinate mapping. From this it is possible to conceive of a browser based system that could do a similar reprojection from the WMTS tiles in EPSG:4385 to the Transverse Mercator base map in EPSG:900913 using WebGL. At the present time, no examples of any systems of this type can be found.

Figures 5.7a, 5.7b and 5.7c show rainfall radar data from the storm of Sunday October 10th 2013. The source code for this example is available from the MapTube GitHub account at the following address: *https://github.com/maptube/meto*.



(a) Sunday 27th October 17:15Z (b) Sunday 27th October 19:00Z (c) Sunday 27th October 21:30Z

Figure 5.7: Rainfall radar data for the storm of Sunday 27th October 2013. Data is taken from the UK Met Office's DataPoint service. Data copyright Met Office.

The class diagram in figure 5.8 shows the basic 'ImageOverlay' class, extended to handle the animated radar sequences shown in figure 5.7a,b,c. This is achieved through

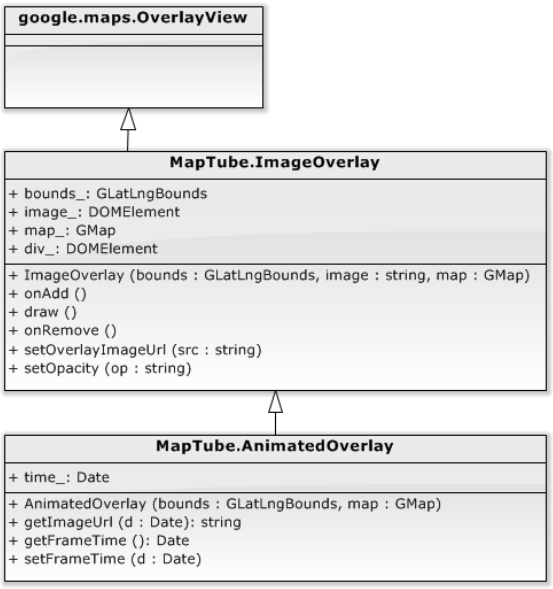


Figure 5.8: MapTube overlay classes for the animated image overlay capable of being added as a data layer on a Google Map.

the 'AnimatedImage.getImageUrl()' method which is overloaded to supply the URL of the relevant RadarNet image frame given its date and time. The animation code on the html page simply takes the start time and adds fifteen minutes to obtain the next frame, looping back to the start when the end time is reached.

Moving on to the synoptic data, due to the quantity and custom display format, a variant of the previously outlined WebGL overlay is used. The requirements for this section are for exploratory data analysis, which implies data handled in its native format and visualised directly. The world 'Global Telecommunications feed' (GTS) contains 1160 synoptic observations, while the UK Met Office's DataPoint site has 121 official stations for the UK, with the feed from their 'Weather Observation Website' (WOW) network containing an aggregate of 1304 worldwide, UK and volunteered community observations. The data contains information on wind speed and direction, temperature, dew point, relative humidity, visibility, gusts, weather type and surface pressure. In order to be able to implement functions like thresholding and de-cluttering, which are necessary for data exploration, the map overlay needs to be able to handle the required complexity of the graphics. Additional functions like contouring are also desirable as, applied to the temperature or pressure fields, this will highlight weather fronts as discontinuities in the pressure field.

The diagram in figure 5.9 shows a standard World Meteorological Organisation station circle containing information for wind speed and direction, total cloud, air temper-

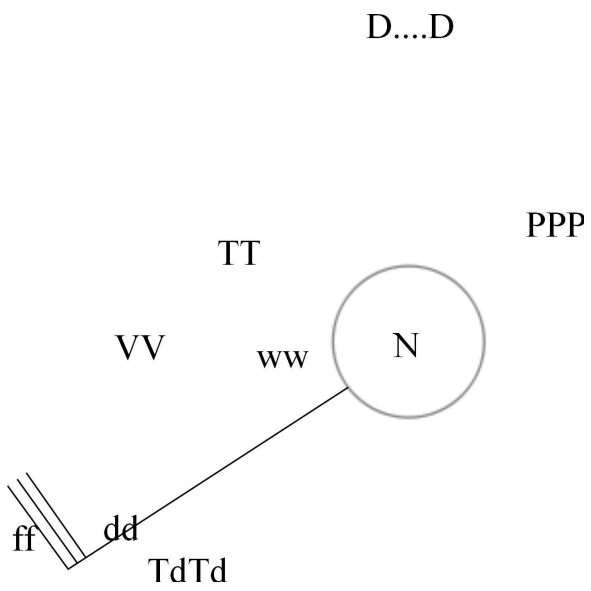


Figure 5.9: World Meteorological Organisation (WMO) Station Circle, METO Form 7, showing wind, total cloud, air temperature, dew point, pressure, visibility and present weather.

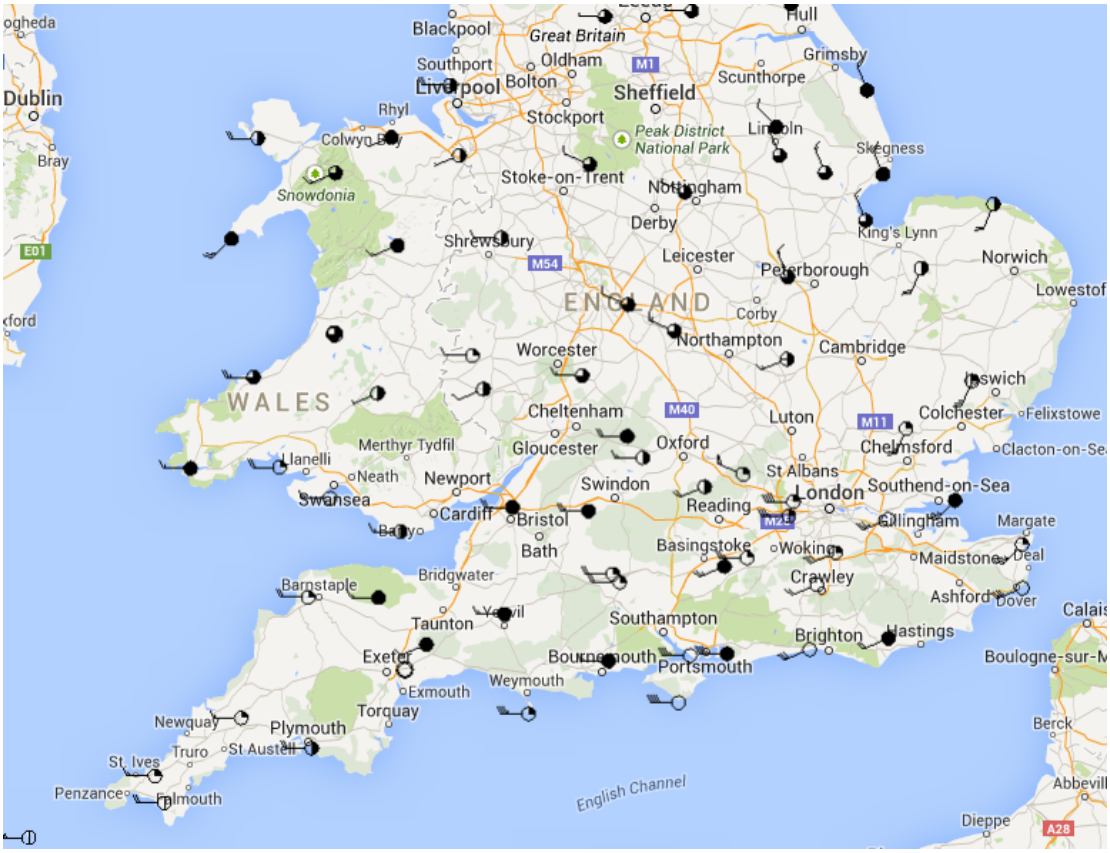


Figure 5.10: Synoptic observation plot for the South of England from 28th October 2013, 0700Z.

ature, dew point, mean sea level pressure (MSLP), visibility and present weather, along with the name of the station or ship making the observation. Using the WebGL system outlined previously for plotting geographic data as a Google Maps layer, this type of synoptic data can be plotted by writing a subclass containing the rendering code. The advantage of using the WebGL system is in the higher quantity of data it can handle and the fact that, with data stored locally, there is a higher level of interactivity. The disadvantage is that the station plot is made using a screen space pixel scale rather than a geographic scale, as the size of the plot does not change as the user zooms in and out. Figure 5.10 shows the synoptic data overlay based on the WebGL code. This library is available for download from: *https://github.com/maptube/meto*.

While the meteorological data viewer is one example of exploratory data visualisation, an obvious extension is to handle the OGC's 'Simple Features' geometry, which is based on points, lines and linear rings forming polygons and holes. By handling general polygons, the framework is now in place for a 'Web GIS' which can load data in from shapefiles and perform similar functions to a conventional desktop GIS. Two approaches exist to the polygon problem: render all the data onto the canvas as a single block, or tile the geometry and reduce the complexity dynamically based on zoom level. The single block of data approach can favour certain types of data visualisation, while the tiling approach favours cartography which covers the entire Earth. The following section completes the WebGL tool kit by adding complex custom geometry using a hybrid approach known as 'vector tiling'.

5.1.4 Vector Tilers

By extending the WebGL visualisation system outlined in the previous section, an alternative technique is to use a method called 'Vector Tiling'. At present, there are two vector tile formats in common use, utilising GeoJSON (Mapnik) and Google Protocol Buffers (MapBox GL). The idea behind vector tiling comes from the cartographic base maps using OpenStreetMap data, where vector data for the map is delivered to the WebGL browser which then renders the tile directly on the client rather than the tile rendering happening on the server. This means that the raw vector data is now available on the client, enabling the map to be styled on a per-user basis locally. When this is extended to maps containing data, the ability to manipulate and style the geometry locally means that the technology has taken the final step towards a true web GIS. This section takes the MapTubeD image tiler, which used late binding of data on the server to create its maps, and adapts it into a late binding vector tiler, where the spatial and aspatial data are delivered to the client separately and joined on the client browser.

The internal architecture of the vector tiler is similar to the *MapTubeD* image tiler, which was part of the ESRC GenESiS project[5] and is not open source. The project has 13,363 lines of code written in C# and uses Microsoft's Windows Communication Foundation (WCF) library to supply the tiler's web service. In order to create an open source project, a new vector tiler project called *MapTubeV*[6] has been started. In architecture terms, the only common elements between the two systems are the tile referencing and geocoding (numerous published algorithms exist), the circular buffer which is central to the performance (a standard design pattern) and the tile caches, which are unique key based disk caches of rendered tiles. The structure of a vector tiler is greatly simplified compared to an image tiler as, without the requirement to render the tiles, the style does not need to be passed to the web service any more and large parts of the code are now redundant.

The design of this web service can be described in one sentence as follows: "Take a shapefile referenced by a unique URI and build a web service that can return the vector data for any of its map tiles based on a quad tree hierarchy". While shapefiles are mentioned here, the idea can be extended to any form of spatial data. The key to performance is the same as for the image tilers, where the first request for a new shapefile blocks all other load requests until it is loaded into the circular buffer, re-

[5] Project: GenESiS, 'Generative E-Social Science' ESRC RES-149-25-1078.
[6] MapTubeV is named for 'V' meaning vector tiler.

projected if necessary and indexed to allow for fast envelope tests. The amount of time taken to download the remote file to the local staging area can not be controlled, so the reprojection task is the next obvious candidate for optimisation. In the code on GitHub, the 'MercatorProjection' class performs the reprojection using ProjNet and the standard GeoAPI interfaces. It was discovered early on that, while extracting all the points from a geometry collection, reprojecting and replacing the points is a simple and effective method, it can also be exceptionally slow for complex geometries. This is because the GeoAPI 'Coordinates' are stored internally for every 'multi' feature and numbered geometry. This means that extracting and replacing coordinates individually, or as a block, requires walking the 'GeometryCollection' hierarchy, which takes time. The recursive implementation used in the code is at least an order of magnitude faster, which is crucial to the vector tiler implementation. The naive implementation has been left in the code base as 'MercatorProjection.ReprojectOldAndSlow()' to enable comparisons to be made. Depending on performance tests, any future improvements are likely to come from increased parallelism, either on the CPU or GPU.

The data flow diagram for the system is shown in figure 5.11, with algorithm 3 listing the processing steps involved. Breaking the system down into basic design patterns, the system is built with a file cache, circular buffer and a library to create vector tiles from a tile bounding box and feature data source. This then forms a crucial architectural element in a distributed WebGIS system. A WebGL enabled browser can now use the MapBox GL JS library to load a shapefile directly by specifying the MapTubeV service as the data source. Taking this a step further, the MapBox code has been modified to include aspatial data attached to the vector data source and 'late joined' with an area key on the point of loading into the browser. This enables the vector tiles to be re-used with different datasets, so, for example, with the 2011 Census at MSOA level, the vector tiles containing the MSOA boundaries can be re-used for every one of the Census tables. This makes for an efficient system for caching boundary data and builds into a web based exploratory GIS system.

The MapTubeV vector tiler code is available on GitHub at the following address: *https://github.com/maptube/MapTubeV*.

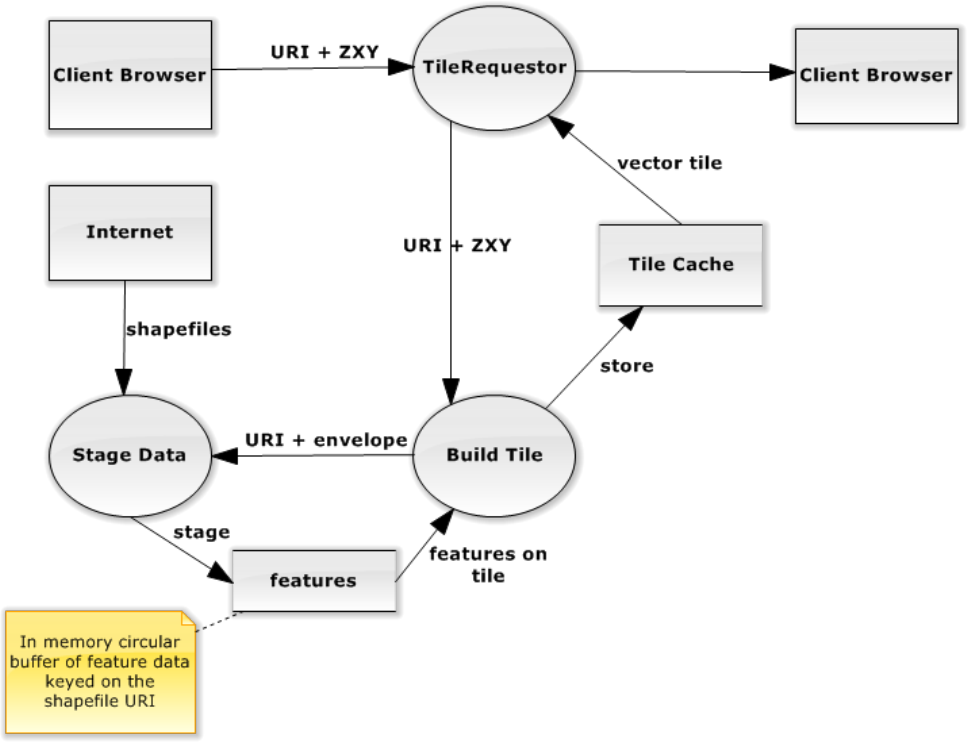


Figure 5.11: Data flow diagram for the MapTubeV system showing the tile requester, caching, building and data staging systems.

Algorithm 3 MapTubeV Vector Tiler

Require: URI, location of shapefile on the Internet
Require: ZXY, tile code Z (zoom level), X and Y numbers uniquely identifying a map tile
Require: $VectorTileCache$, file cache of vector tiles keyed by URI and ZXY
Require: $FeatureBuffer$, circular buffer of feature data keyed by URI

if Tile ZXY for URI is in $VectorTileCache$ **then**
  $tile \leftarrow$ `cached tile from` $VectorTileCache$
  return $tile$
else
  if Dataset URI **not** in $FeatureBuffer$ **then**
    Download .shp, .dbf and .prj files from URI and stage locally
    Reproject shapefile into Spherical Mercator
    Store reprojected features in $FeatureBuffer$
  end if
  $envelope \leftarrow ZXY$ `tile code converted into geographic bounding envelope`
  $features \leftarrow$ `all features intersecting` $envelope$ `from` $FeatureBuffer$ `for` URI
  $tile \leftarrow$ `new vector tile`
  for all f in $features$ **do**
    clip f to $envelope$
    simplify f to appropriate scale for Z
    add f to $tile$
  end for
  return $tile$
end if

5.1.5 Conclusion

This example was based around exploratory data analysis where there are multiple variables and multiple points in time. Data is on the desktop due to the interactive nature of exploration. Building a similar system using the tiled map approach would limit the level of interactivity due to the data being on the server while the interaction happens on the client. By using the 'fat client' architecture, the WebGL technology, which was intended for browser based games, is re-purposed for geographic rendering. While this allows for more powerful visualisation, the server is still required for functionality that a web browser cannot handle, for example loading binary format data like shapefiles. This development path follows the 'CyberGIS' approach favoured by Wang [Wan10] with a framework of web services providing loading and analysis functions. Scalability is achieved with cloud computing, with libraries like 'Geotools' running on Hadoop clusters. While this is a valid approach, problems still exist with moving large amounts of data to and from the cloud, which is why cloud-based systems have not been mentioned directly here. The weather examples use real-time data hosted on Microsoft's Azure platform, so the cloud is being used implicitly as the source of the information. Data exploration and inter-operability are the two aims of this example, with 'Web GIS' used to explore real-time data that originates on the web, rather than using a conventional desktop GIS system which expects datasets in the form of files.

5.2 Example 2: Data Stores

Data stores like the 2011 Census and *data.gov.uk* contain large amounts of data about cities. How can the techniques explored in previous chapters be applied to extracting information from these sources? The following quote from Gibson, written in 1984, describes his imagination of stores of data resembling cityscapes. The research in this section finds structure in contemporary stores of data, under the hypothesis that this leads to further understanding.

> "Cyberspace. A consensual hallucination experienced daily by billions of legitimate operators, in every nation, by children being taught mathematical concepts... A graphic representation of data abstracted from the banks of every computer in the human system. Unthinkable complexity. Lines of light ranged in the nonspace of the mind, clusters and constellations of data. Like city lights, receding..." (William Gibson, Neuromancer, [Gib95])

What does a data store look like?

Taking the UK Census 2011 bulk uploads as an example[7], a WebGIS system can be built from the existing components to visualise all the data. Previously, the MapTube, London Profiler or Census Profiler projects would have created each map individually, with the data comparable as 'mash-ups' by overlaying different layers ([Bat+10a] and [Gib+08]). MapTube is different to the other systems in that it aims to make it easier for the general public to make maps from data and upload them to the site, becoming a source of data in itself. By simplifying the process of making maps through intelligent data processing algorithms, the work flow is there to be able to take a data store and automatically mine it for spatial information. By taking the entire set of data as a whole instead of individual datasets, relationships between the data can be investigated. This project is called the 'Data Store Miner' and uses the MapTube 'Geometry Finder' library and web service to identify the spatial context of the data, along with the WebGIS techniques created in the previous section. In addition to this, a secondary aim is to make it easier to navigate data stores using a form of 'search analytics' where links between geospatial datasets become 'feature paths' to be followed. Using the keyword 'population' to filter the search results on the home page of the MapTube website yields a large number of very similar maps (figure 5.12).

[7] Census 2011 bulk download: *https://www.nomisweb.co.uk/census/2011/bulk/r2_2*.

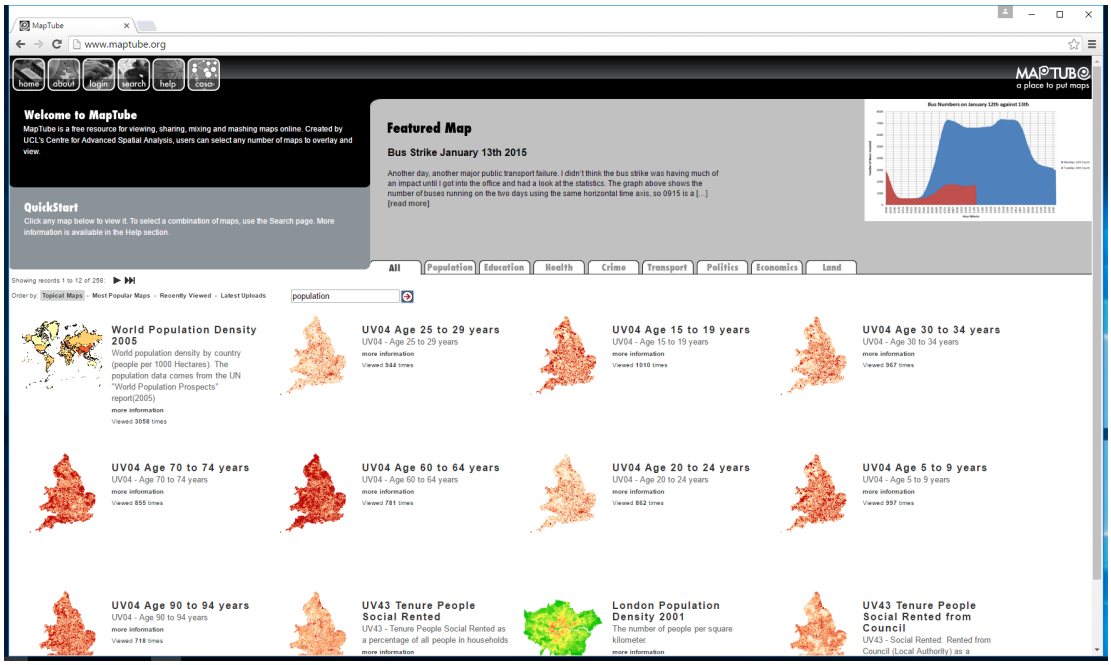


Figure 5.12: The home page of the MapTube website *www.maptube.org* with the results filtered using the keyword *population*. The result is many similar looking maps.

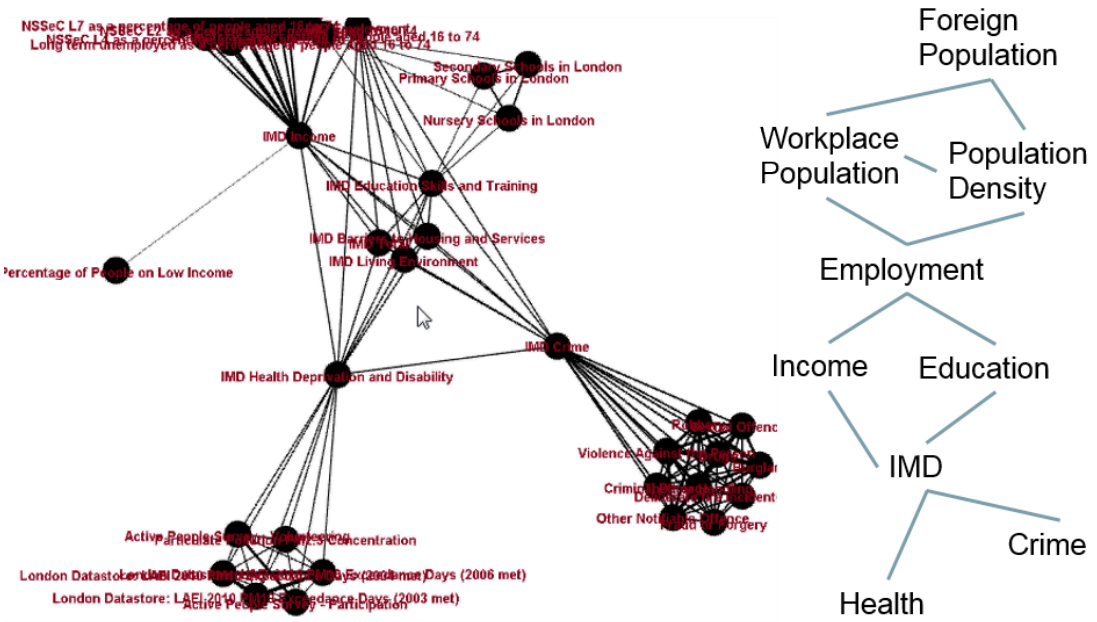
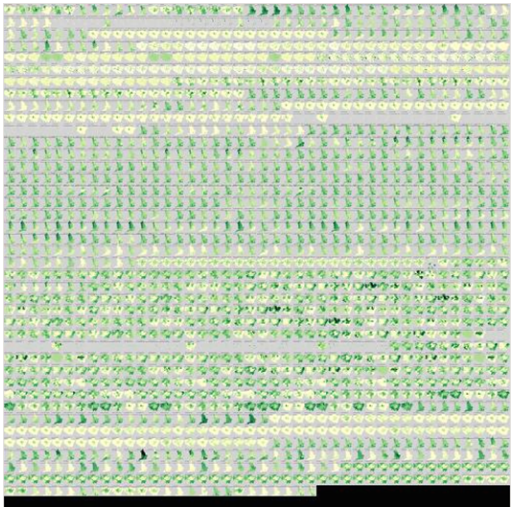
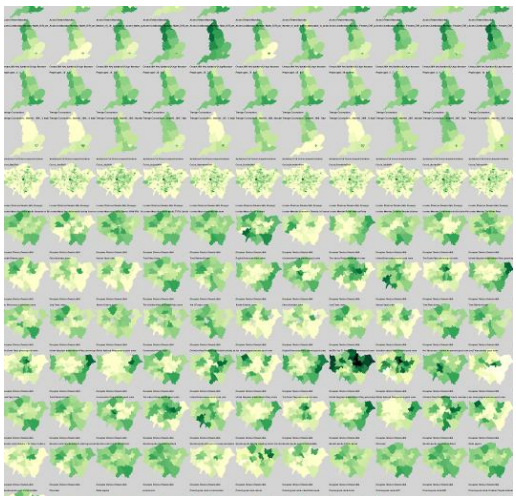


Figure 5.13: Keyword graph built from the MapTube website with a simplified annotated version shown on the right.

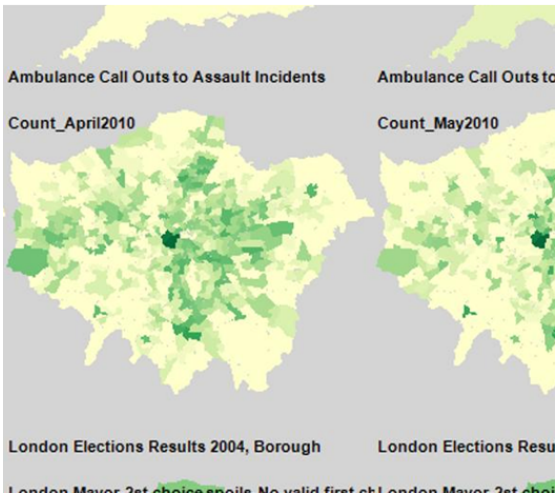
Even with the thumbnail previews of the maps, the 258 returned results are difficult to navigate, being mostly 'England and Wales' shaped maps of Census data. The question is how they all relate to one another and which are more important? Given the large amount of information being stored here, the aim is to give structure to the data, which enables it to be navigated more easily and relationships discovered. Figure 5.13 shows the structure of how some of the maps on the site relate to one another. This is compared with the view of all the possible Census maps shown in figures 5.14a, 5.14b and 5.14c.



(a) All maps.



(b) Zoomed in closer.



(c) Individual maps.

Figure 5.14: All of the England and Wales maps which can be built automatically from MapTube. The maps are coloured using a quintile classification.

This example builds on the previous 'WebGIS' map making example as it is a work flow to source large quantities of geospatial data from Internet data stores automatically in order to use it in a spatial data mining work flow.

The input to the 'Data Store Miner' is a manifest file and semantics containing information about where to download the data from. This is then passed to the 'Geometry Finder' for identification, before being built into a web based map using the mapping system described previously. Figure 5.15 shows the data flow diagram.

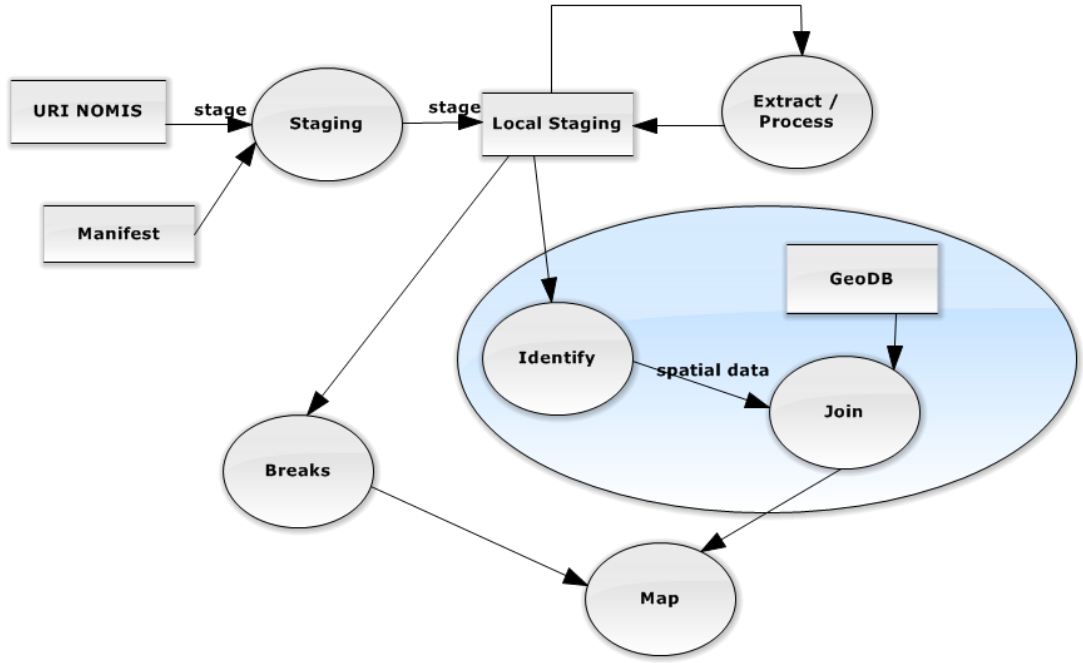


Figure 5.15: Data Flow Diagram for Automatic Mapping.

The 'Geometry Finder' is part of the MapTubeD project and is not open source, so can only be exposed as a web service, as described in the following table 5.3.

http:\\shoal.casa.ucl.ac.uk\GeometryFinderHandler.ashx?r=[g|p|k](&u=...)(&g=...)	
QueryString	Description
r=g & u=DataURI	Search csv file identified by data URI for area geometry
r=p & u=DataURI	Search csv file identified by data URI for point geometry
r=k & g=BoundaryName	Return all area keys belonging to the named boundary dataset e.g. GOR for Government Office Regions

Table 5.3: MapTubeD Geometry Finder Web Service.

The result from passing a CSV file to the Geometry Finder web service is an XML file containing column probabilities and geometry types which can be used to decide which geometry to join the CSV file with and which column contains the most probable area key to join on. This highlights why this is a web service and not a library as it relies on a large geocoding database stored on the server which needs to be maintained.

Building on the 'Geometry Finder' and 'Data Store Miner' projects, data can be sourced from the Internet, identified and joined with the appropriate boundary file if needed. In the context of the Census 2011 example, along with automatically making maps from the data, with a slight modification the software can perform spatial correlation to see how all the variables relate to one another. Figure 5.16 shows the modification to the automatic mapping system to perform spatial cross correlation of variables in datasets.

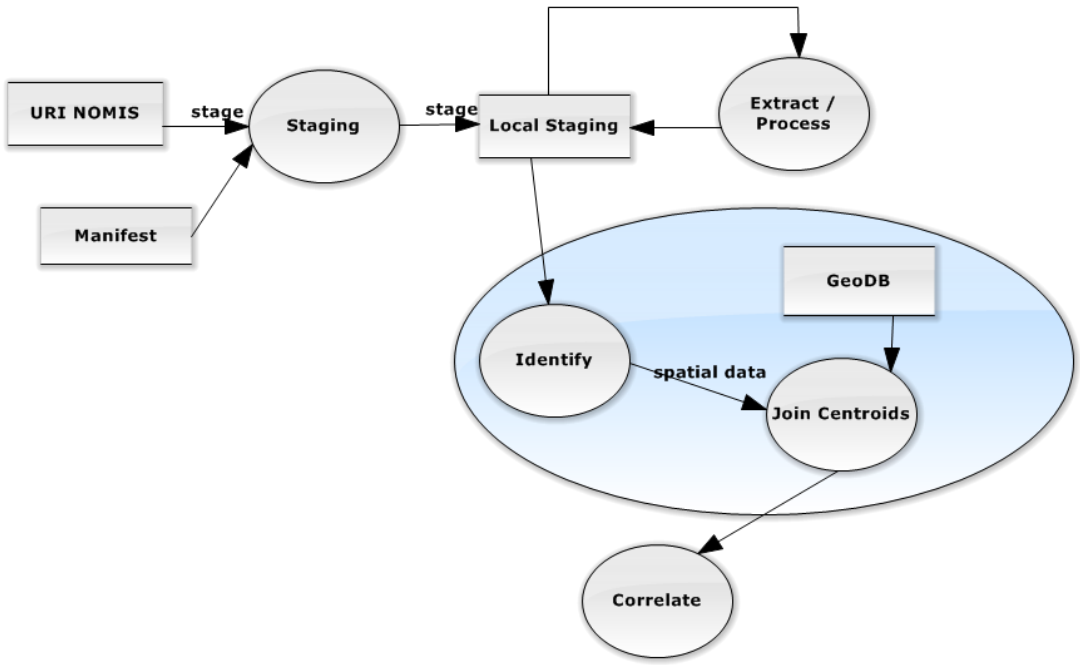


Figure 5.16: Data Flow Diagram for Spatial Cross Correlation.

This is a desktop GIS application that pulls data from an Internet data store and performs all processing as a batch on the local machine. While this is adequate as a proof of concept, the use of cloud computing to improve the speed of the processing is an obvious next step once the method has been proven. The need for either cloud computing or GPGPU optimisation becomes clear when the algorithm is examined in more detail. Using the table of Census variables from appendix A, there are 2,558 separate sets of variables that need to be correlated. If this is visualised as a square correlation matrix

with 2558 x 2558 cells, then that amounts to 6,543,364 dataset correlations. In practice, only the upper triangle is required as correlation is commutative, so the number of iterations can be expressed as follows:

$$T = \frac{(N+1)N}{2} \tag{5.3}$$

Putting $N = 2558$ into equation 5.3 gives the number of required tests as $T = 3,272,961$. Algorithm 4 shows the methodology used to stage all the required files and run the correlation test function T times.

Algorithm 4 Data Store Miner Correlation

Require: $catalogue$, list of datasets and URIs
Require: $variables$, list of variables contained in the datasets and meta-data
1: $N \leftarrow$ `number of files in` $catalogue$
2: **for** i=0 **to** N **do**
3: Stage file i locally
4: **for** j=i **to** N **do**
5: Stage file j locally
6: **for all** variables V_i in file i **do**
7: **for all** variables V_j in file j **do**
8: SpatialCorrelate(V_i,V_j) {Correlate function replacing "MakeMap"}
9: **end for**
10: **end for**
11: **end for**{for j}
12: **end for**{for i}
13: **return**

Here, the $SpatialCorrelate(V_i, V_j)$ function is a spatial cross correlation between two datasets, i and j, based on similarity of values and distance. Equation 5.4 shows the calculation of the correlation value, I, where X and Y denote the datasets V_i and V_j from algorithm 4. The (i, j) indices in the spatial correlation function represent the comparison of every area against every area, while in algorithm 4, (i, j) represent every dataset correlated against every other dataset. It is this nesting of combinatorial flow at the two algorithmic levels which causes performance problems as the numbers of datasets and number of geographic areas increases.

$$I = \frac{1}{S_0} \sum_i \sum_j \frac{(Y_i - \mu_y)}{\sigma_y} \times W_{ij} \times \frac{(X_j - \mu_x)}{\sigma_x} \tag{5.4}$$

$$S_0 = \sum_i \sum_j W_{ij} \tag{5.5}$$

I	Spatial cross correlation coefficient between X and Y, $I = [-1..1]$
i, j	area index in datasets X and Y where $i = j$ denotes the same area
X, Y	datasets to be tested for correlation
μ_x, μ_y	mean of all values in the dataset
σ_x, σ_y	standard deviations of X and Y
W_{ij}	$\frac{1}{distance(km)}$ or 1 if $i = j$

Algorithm 5 shows the pseudo-code for how this spatial correlation is performed. The main point to notice is that the algorithm is of order $\mathcal{O}(A^2)$. This is a problem because A in this instance is the number of geographic areas in each of the two datasets being correlated, which are available for the following four geographies: Wards ($A = 8588$), Middle layer Super Output Area (MSOA $A = 7201$), Lower layer Super Output Area (LSOA $A = 34753$) and Output Area (OA $A = 181408$).

Algorithm 5 SpatialCorrelate

Require: X, list of first dataset values
Require: Y, list of second dataset values
Require: C, list of centroid points
Ensure: lists X, Y and C all match spatially, so $X[i]$, $Y[i]$ and $C[i]$ all reference the same spatial area
Ensure: length(X)==length(Y)==length(C)
1: $A \leftarrow length(X)$
2: $I \leftarrow 0$
3: **for** i=0 **to** $A - 1$ **do**
4: Stage file i locally
5: **for** j=i **to** $A - 1$ **do**
6: $w_{ij} \leftarrow value \propto distancebetween(C[i], C[j])$
7: $I = I + (X[i] - \mu_X)/\sigma_X \times (Y[j] - \mu_j)/\sigma_Y \times w_{ij}$
8: **end for**
9: **end for**
10: **return** I

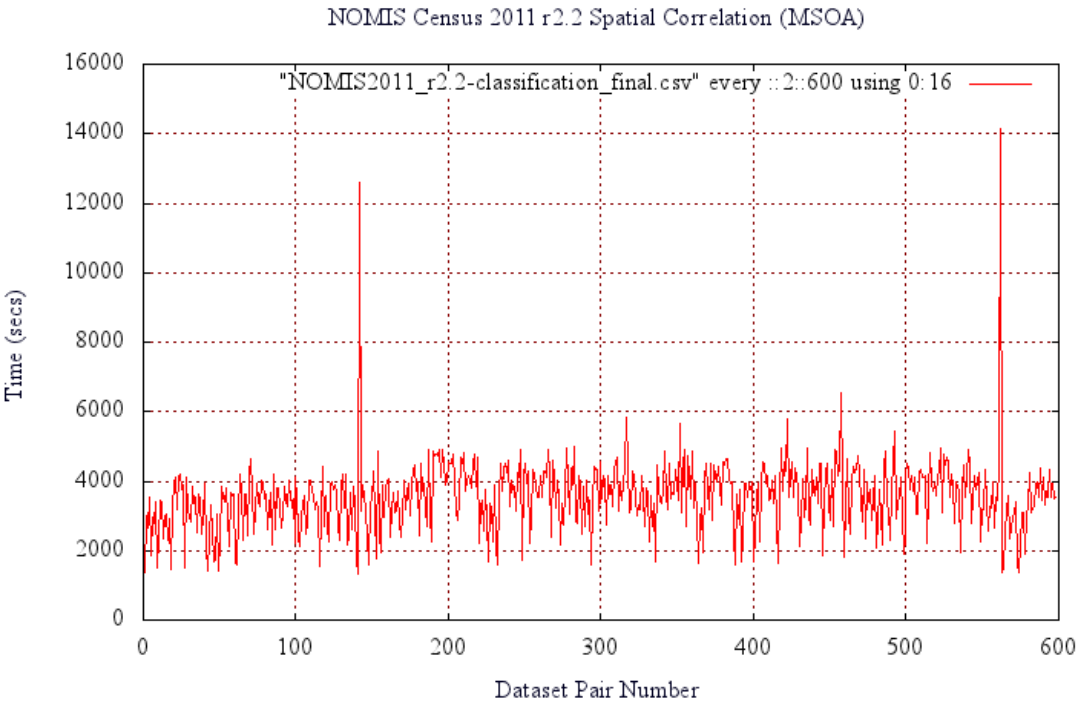


Figure 5.17: Census 2011 Spatial Correlation Timings.

Preliminary results show the amount of time taken to perform each correlation on the data at MSOA level ($A = 7201$). Figure 5.17 shows the first 600 lines of the log file plotted with time in seconds on the y axis. Each point on the x axis is a single correlation, so the y axis shows the time taken for each one. The two peaks at $x = 145$ and $x = 565$ show where data staging is taking place, with the data being downloaded from the data store, unpacked and processed. Table 5.4 shows a small portion of the data used to create this graph.

Using the 30,696 rows of data from the results table, the mean, excluding the staging delays, is calculated as 2,299 milliseconds. This initial dataset only contains the results of the variables from the first table (KS101, population) correlated against every other variable in all the other tables. There are 104 datasets which totals 7.56GB in the staging area (unzipped, all four geographies). Based on this preliminary data, the calculation of the full spatial correlation matrix for all 2,558 variables is likely to take about 87 days.[8]

[8]The calculation based on $87days \approx 3272961 \times 2.299s/3600/24$.

	I	$File_i$	$File_j$	Var_i	Var_j	$VariableName_i$	$VariableName_j$	ms	deltams
1	0.997924877	0	0	1	1	KS101EW0001	KS101EW0001	8261	0
2	0.986228992	0	0	1	2	KS101EW0001	KS101EW0002	9677	1416
3	0.985908516	0	0	1	3	KS101EW0001	KS101EW0003	11081	1404
4	0.972168253	0	0	1	4	KS101EW0001	KS101EW0004	12449	1368
5	0.225975799	0	0	1	5	KS101EW0001	KS101EW0005	15439	2990
6	0.307325901	0	0	1	6	KS101EW0001	KS101EW0006	18108	2669
7	-0.037374855	0	0	1	7	KS101EW0001	KS101EW0007	21650	3542
	...								
30687	-0.012061773	0	103	12	86	KS101EW0012	CT00100086	18237062	3529
30688	-0.028926019	0	103	12	87	KS101EW0012	CT00100087	18240167	3105
30689	0.044596786	0	103	12	88	KS101EW0012	CT00100088	18243321	3154
30690	0.096536754	0	103	12	89	KS101EW0012	CT00100089	18246185	2864
30691	0.070795003	0	103	12	90	KS101EW0012	CT00100090	18248159	1974
30692	0.126514217	0	103	12	91	KS101EW0012	CT00100091	18251526	3367
30693	0.026441067	0	103	12	92	KS101EW0012	CT00100092	18254622	3096
30694	0.085927621	0	103	12	93	KS101EW0012	CT00100093	18257613	2991
30695	0.033316855	0	103	12	94	KS101EW0012	CT00100094	18260812	3199
30696	0.016183034	0	103	12	95	KS101EW0012	CT00100095	18263381	2569

Table 5.4: Correlation results from the data store miner. The "I" value is the correlation result while "File_i" and "Var_i" represent the index of the first table, "File_j" and "Var_j" represent the index of the second table. The plain text variable names allow for identification using the Census table lookup file which describes the ONS methodology used in creating the table. Finally, the number of milliseconds the program has been running and the delta between the last two values give the amount of time it takes to calculate each correlation point.

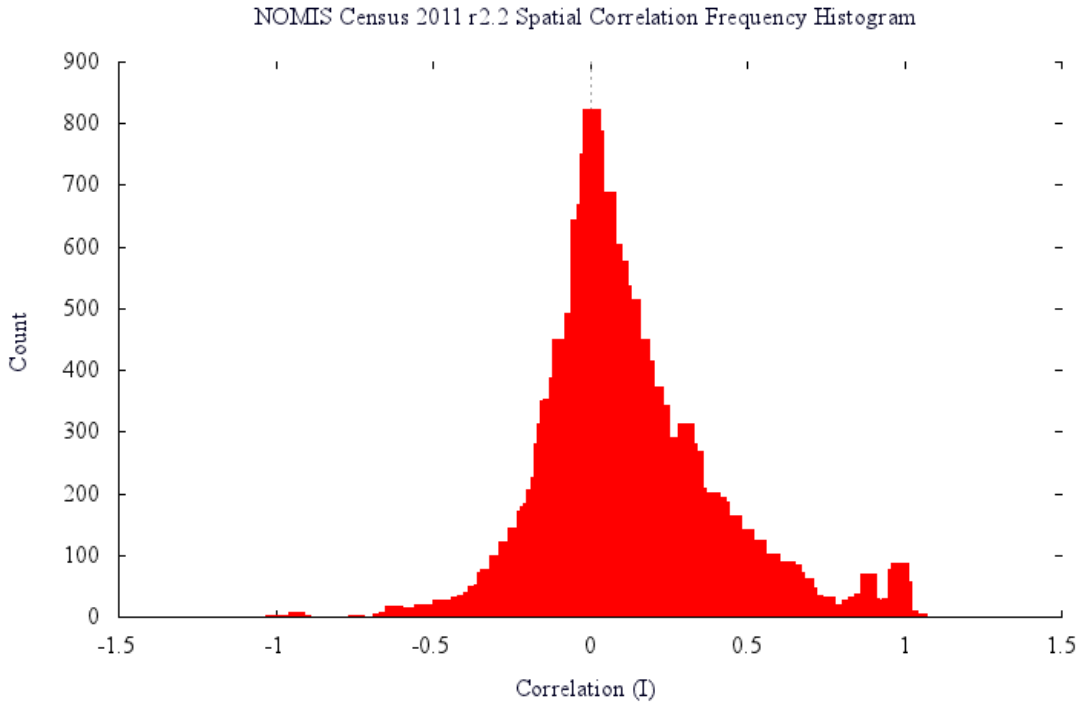


Figure 5.18: Census 2011 Spatial Correlation Frequency Histogram.

The graph in figure 5.18 shows a frequency histogram of the correlation values (I) using a bin width of 0.01. The mathematical formulation of the correlation value, I, limits the range to ± 1.0. The data is investigated further in the following sections, but the peaks around $+1.0$ are caused by certain population variables being repeated in multiple datasets, causing $I = 1.0$ on a frequent basis.

On a final note, release 2.2 of the Census 2011 data from NOMIS was chosen specifically because of the regular structure of data across all of the tables and columns. Some data can be problematic though, for example the Census migration statistics in the "Origin-Destination" tables release. A subset of the travel to work table *"wu01ew_msoa"* is shown in table 5.5 to demonstrate the structure.

Line	Area of Residence	Area of Workplace	All categories: Sex	Male	Female
1	E02000001	E02000001	1506	976	530
2	E02000001	E02000014	2	2	0
3	E02000001	E02000016	3	2	1
4	E02000001	E02000025	1	0	1
5	E02000001	E02000028	1	1	0
6	E02000001	E02000051	1	0	1
7	E02000001	E02000053	2	0	2
8	E02000001	E02000057	1	0	1
9	E02000001	E02000058	1	1	0
...	...	...	...	...	...
392	E02000002	E02000001	74	39	35
393	E02000002	E02000002	73	18	55
394	E02000002	E02000003	53	20	33
395	E02000002	E02000004	5	2	3
...	...	...	...	...	...
822	E02000003	E02000001	212	110	102
823	E02000003	E02000002	16	6	10
824	E02000003	E02000003	203	67	136
825	E02000003	E02000004	9	4	5
...	...	...	...	...	...

Table 5.5: Table "wu01ew_msoa" from the NOMIS Census Bulk upload "rOD1", available from https://www.nomisweb.co.uk/census/2011/bulk/rOD1. The full table contains 2,402,202 lines.

In order to use this travel to work data it has to be aggregated into either numbers of commuters at every origin location, or at every destination location. This method is similar to a SQL "GROUP BY" clause, aggregating origin and destination flows into a representation that allows for correlation with the other data.

5.2.1 Correlation in the Text Domain

The scale of the Census data, containing 5,460 table combinations and 3,272,961 variable combinations, along with the fact that tables like 'Country of Birth (*KS204*)', 'Passport Held (*KS205*)' and 'Household Language (*KS206*)' are highly correlated, makes it hard to identify any interesting or unusual facts about the data. This is not strictly a case of identifying outliers, because some of the data is highly correlated, being related for obvious physical reasons. Analysis is complicated by the need to find relationships in a quantity of data populated with already well-known causal relationships. What is required is a further technique to identify the interesting data hidden in amongst the obvious.

In addition to analysing the Census data itself, a text mining process can be used as an additional measure by making use of the description documents which exist for every table. The technique used is to create a 'bag of words' (BOW) for each document describing one of the Census tables. A PDF document for each table is available online[9] and can be downloaded in bulk using the same *http* staging method that the DatastoreMiner uses for the data tables. In effect, the whole data mining process is repeated, but with a parallel set of data about the 2011 Census from a different domain. Now, both a data domain representation and a meta-data domain representation can be analysed together. The flowchart in figure 5.19 shows the methodology used for creating the 'bag of words' representation.

Once the data has been assembled, a 'Keyword Correlation' function is performed as shown in the flowchart. Due to the nature of the algorithm, this requires the bag of words representation for every document to be held in the computer's memory as the 'Term Frequency Inverse Document Frequency' (TFIDF) algorithm needs to act over all the words in every document. Fortunately, the number of words required is comparatively small. The first steps shown in the flowchart are to extract all the words from the PDF using the open source 'iTextSharp' library, followed by splitting the document on white space characters to separate individual words and to build a histogram of the number of occurrences of each word. In addition to this, a small amount of filtering is performed to discard any words of three characters or less and any other non-text symbols like numbers and punctuation. Also, a *stop words* list is used to remove words which are not deemed to be useful in text matching. In this instance, the 'Glasgow Stop

[9]The url is: *https://www.nomisweb.co.uk/census/2011/{TableName}.pdf* where {TableName} is the name of the required table, for example, ks101ew.

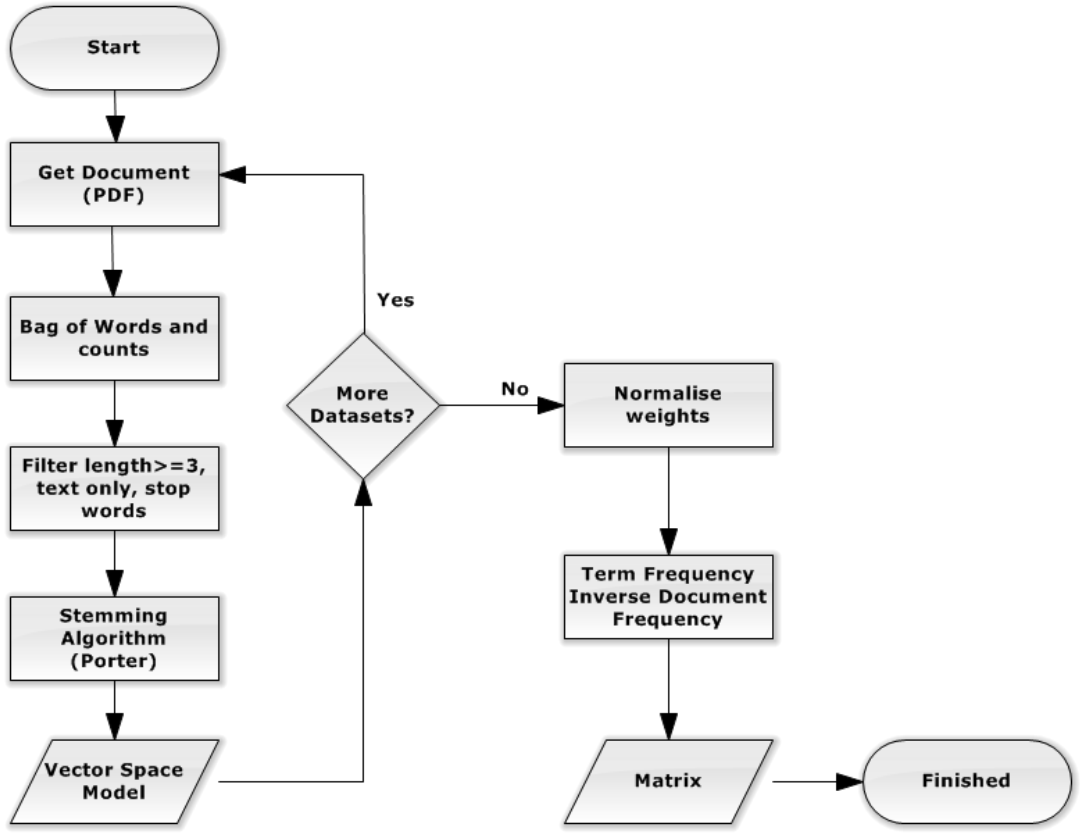


Figure 5.19: Flow diagram for the Data Store Miner natural language processing of documents.

Words' list[10] was used, which contains 319 words considered useless in text classification systems. Finally, a stemming algorithm is used to identify the morphological roots of words like 'computer', 'computers' and 'computing', reducing them to the common stem of 'comput' in order for them to be recognised as a single concept. The stemming algorithm used here is the Porter Stemming algorithm [Por80], with the word counts in the final vectors normalised to give a total length of 1 for each document.

One problem was encountered during this process, which is endemic to the data. The tables *KS601*, *KS602*, *KS605*, *KS606*, *KS608*, *KS609*, *KS611* and *KS612* do not have separate description documents due to how the data is related. Table 5.6 shows how the documentation is structured.

PDF Document	Tables Covered
ks603ew.pdf	KS601, KS602, KS603
ks607ew.pdf	KS605, KS606, KS607
ks610ew.pdf	KS608, KS609, KS610
ks613ew.pdf	KS611, KS612, KS613

Table 5.6: Census PDF description documents and their associated tables.

[10]For the Glasgow Stop Words list see: *http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words*.

Initially, the correlation coefficients for all the tables were calculated based on the sum of products of the weights, which is the frequency of shared words. Due to the fact that the description documents are all based on the same text template, with only selected paragraphs relating specifically to the table in question, this resulted in a very high correlation between all the documents. Figure 5.20 shows the resulting distribution of correlation results.

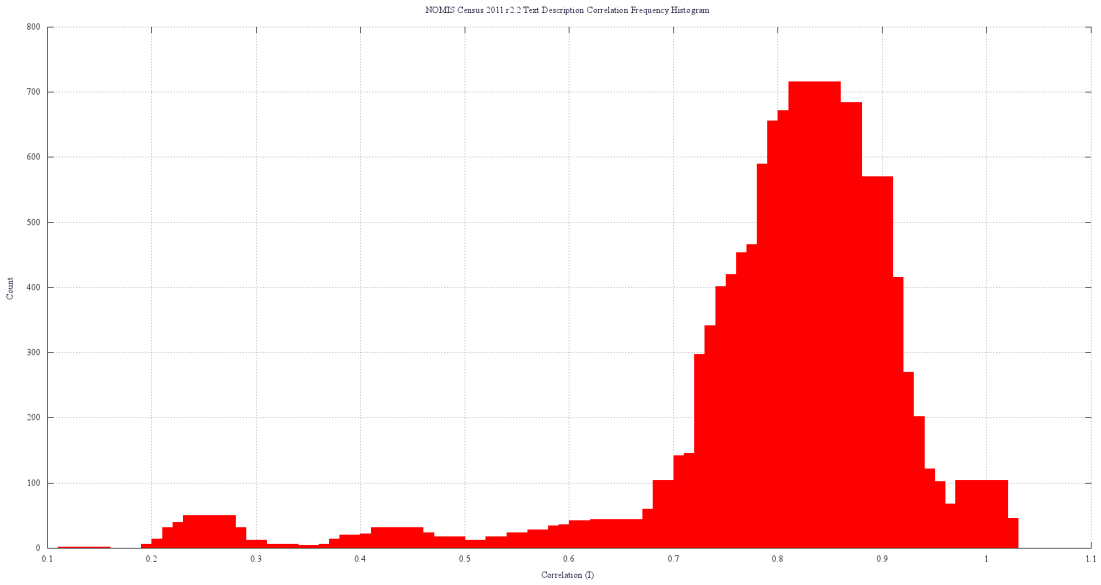


Figure 5.20: Frequency histogram for text correlation values, $n = 104 \times 104 = 10816$, $\mu = 0.80$, $\sigma^2 = 0.013$, $\sigma = 0.12$.

In order to compute a more realistic correlation metric, the 'Term Frequency Inverse Document Frequency' (TFIDF) system outlined in [Bra07, pp243, ch15.5] was used. This is based on the product of two weights, defined as how frequently a word appears in a document and the inverse of how frequently the word appears in all the documents. The idea behind this approach is to increase the weight of a word which occurs more frequently in the document under examination, while reducing the weight of a word that appears in many documents and so has reduced information associated with it.

The 'Term Frequency' (W_{TF}) is the count of the number of times a given word, or 'term', t_j, appears in the document:

$$W_{TF} = count(t_j) \tag{5.6}$$

The 'Inverse Document Frequency' (W_{IDF}) of t_j is defined as:

$$W_{IDF} = log2(n/n_j) \tag{5.7}$$

Where n is the number of documents and n_j is the number of documents containing tj. Then the 'Term Frequency Inverse Document Frequency' weight (W_{TFIDF}) is defined as the product of the two weights:

$$W_{TFIDF} = W_{TF} \times W_{IDF} \tag{5.8}$$

After applying the TFIDF algorithm, the separation between the correlated and uncorrelated datasets in terms of their description similarity can be seen from the frequency histogram of I_{text} correlation coefficients plotted in figure 5.21. The correlation pairs approaching $I_{text=1.0}$ can be discarded if the corresponding data correlation, I_{data}, is also close to 1.0. By plotting this as a second axis on a 2 dimensional scatter plot, the situation starts to become clearer.

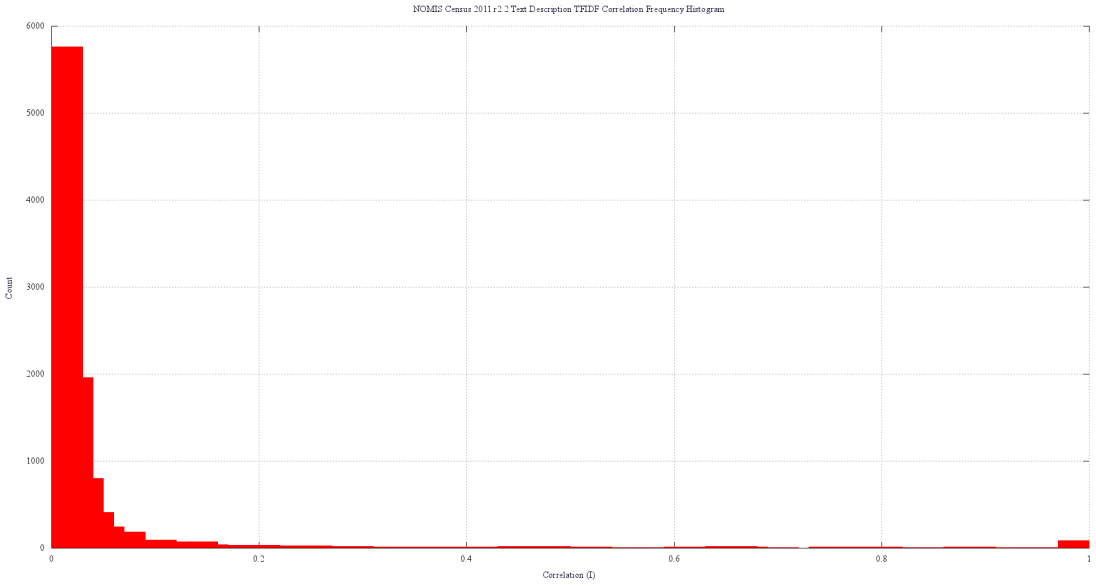


Figure 5.21: Frequency histogram for text correlation values using TFIDF algorithm, $n = 10816$, $\mu = 0.058$, $\sigma^2 = 0.028$, $\sigma = 0.17$.

Figure 5.22 shows the data domain correlation plotted against the text domain correlation using simple word frequency histograms. Each point on this diagram represents a single dataset pair with an (I_{data}, I_{text}) correlation pair used to plot the point as (X, Y). There are no negative text correlation values, as this is impossible due to the correlation value being derived from the count of word occurrences. For this reason, only two quadrants of the graph contain data. Also visible from the graph is the fact that the

text correlation values, I_{text}, are biased towards the upper end of the scale, as demonstrated by the large clustering above the line $I_{text} = 0.7$. This is a visual confirmation of the fact that the documents contain similar plain English words, interspersed with infrequent, but vitally important, technical information.

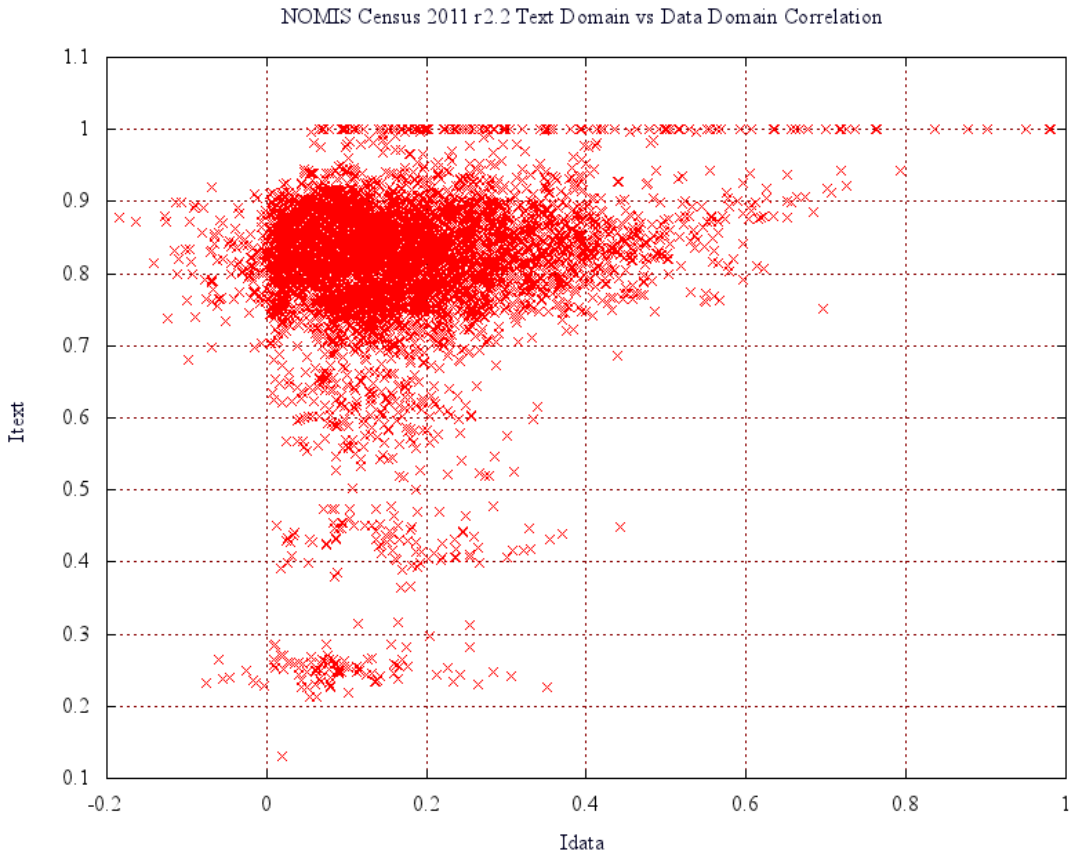


Figure 5.22: NOMIS Census 2011 r2.2 Text Domain (Itext) vs Data Domain (Idata) Correlation. This matches figure 5.20.

The graph in figure 5.23 shows the effect of removing words using the TFIDF algorithm. Now, the words are biased to give infrequent words more value, effectively screening out words that have a low information content as they appear in all the documents. The difference between the graphs in figures 5.22 and 5.23 show that using the TFIDF algorithm has transformed the distribution of the I_{text} values, which now contains a cluster below the horizontal $I_{text} = 0.1$ line.

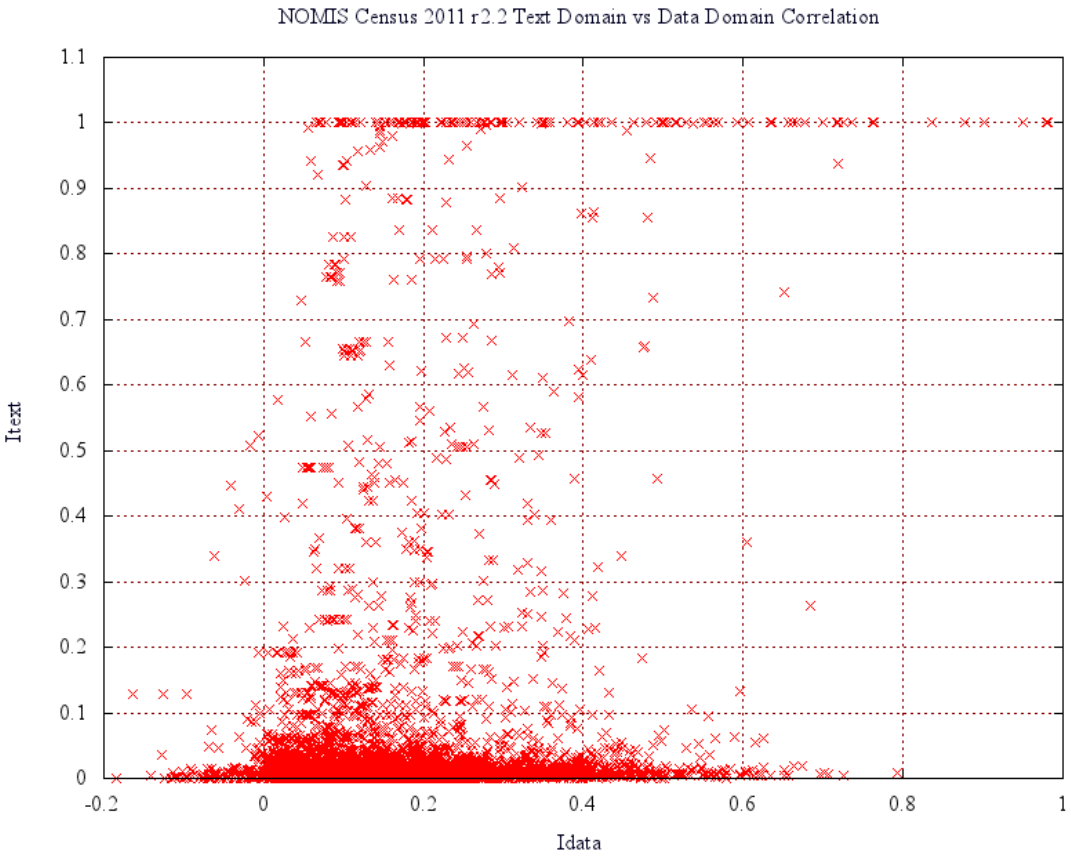


Figure 5.23: NOMIS Census 2011 r2.2 Text Domain (Itext) vs Data Domain (Idata, TFIDF) Correlation. This matches figure 5.21.

The diagram in figure 5.24 shows the areas in the two upper quadrants most likely to contain interesting data. These are defined as where the I_{text} and I_{data} correlation values are mismatched. This can be defined as minimum distance from the line $Y = X$, or as an 'exclusive-OR' function.

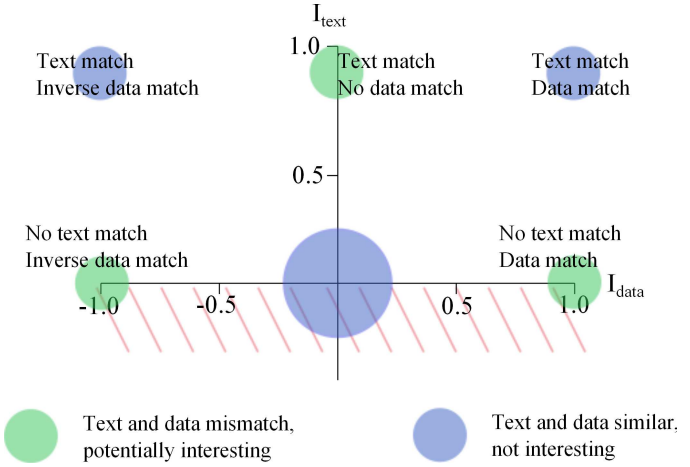


Figure 5.24: Diagram showing where data to be investigated further is likely to be located on a plot of I_{data} vs I_{text}.

One method of visualising the table relationships is to define a new variable as the distance, d, of a correlation pair, (I_{data}, I_{text}), from the 45 degree line. In this example, all of the data is in the positive I_{text} region, while I_{data} has a range of $[-0.2, 1.0]$. With the majority of the data in the positive quadrant, the minimum distance from a point to the line $P = \lambda(xi + yj)$ distance can be defined as follows:

$$d = \left\| \frac{y - x}{2} \right\| \tag{5.9}$$

If the other three quadrants are required, then the formula can be changed to take the minimum distance to the other three 45 degree lines $(xi - yj)$, $(-xi + yj)$ and $(-xi - yj)$. An alternative method is to normalise the correlation pair's vector and calculate the angle deviation from 45 degrees using the dot product, but that technique has not been used in this example.

5.2.2 Pattern Matching

While the spatial cross-correlation technique provides one possible similarity metric, it does not satisfy the requirement for stream mining as every pair of maps needs to be compared. One alternative is to classify the maps into types using a pattern matching algorithm, so a single run through the data using linear feature detection classifies every map as belonging to one of a number of classes. If the degree of match for every class is included, then this additional information can be used as an indicator of possible relationships or, alternatively, that the number of classes needs to be increased. If a map does not match any of the classes to a high enough degree, then this suggest a new group. Techniques like self organised feature maps (SOFM) from neural networks, linear feature detectors and clustering techniques like k-means clustering all fall into this category.

A related technique from stream mining involves the 'streaming k-means' algorithm, outlined by Ted Dunning, Chief Applications Architect for MapR[11]. This uses a single pass through the data to create lots of approximate cluster centroids, which are then clustered using a high quality clustering algorithm to compute the final centroids. These techniques are based on the application of Zipf's law and the quantity of data being processed. Zipf's law states that, "the frequency of a data element is inversely proportional to its rank in the frequency table" [Zip49]. Because of these factors, obtaining an approximate result in a reasonable amount of time, using one pass through the data, is desirable to obtaining an accurate result over a much longer time. Algorithms like 'HyperLogLog', 'CountMin' and 'streaming k-means' all make use of hashing functions and the relatively low probability of hash collisions, as predicted by the inverse frequency relationship of Zipf's law, to process the data. This idea of using hash functions to separate data into groups can also be applied to the map correlations in the previous section. If the maps, which are vectors of 7201 dimensions, are partitioned into a small number of groups where every map belongs to zero or more groups, then relationships between the maps can be detected more easily. In real terms, most Census data is related in some way to population and population density, so any maps that follow the underlying population trend form an obvious first group. Then additional factors like age structure, education, language, employment, transport, housing and living arrangements can also be proposed.

[11] See: *https://www.mapr.com/blog/some-important-streaming-algorithms-you-should-know-about.html* (11 September 2015).

In order to build a feature detector for maps, the existing Census 2011 dataset and previous correlation data can be used as a training set. By looking at the map clustering as a pattern matching problem, the Kohonen self organised feature map described in [Hay94, Ch10.5] is used to reduce the dimensionality to a pair of (X,Y) coordinates on the feature surface. In this example, a grid of 4x4 neurons are trained using the competition algorithm of Kohonen, forming a set of feature detectors. The 2,558 Census variables are presented to the network in a random sequence with the raw inputs normalised to remove any magnitude bias. An identical method of normalisation to the previous correlation from equation 5.4 is used where the raw data value is subtracted from the mean and divided by the standard deviation. The normalised data is then presented to the input of every competition neuron in the network, with the neuron with weights closest to the input being reinforced by adding the error times the learning rate to each weight. In addition to the learning rate, a neighbourhood function is used, so any output neurons in the vicinity of the winner are also reinforced. This brings the weights of the winning neuron and any within its immediate vicinity closer to the input pattern presented, forming a cluster and causing the output grid to self-order and specialise, so neighbouring neurons respond to similar input patterns and far away neurons respond to different input patterns.

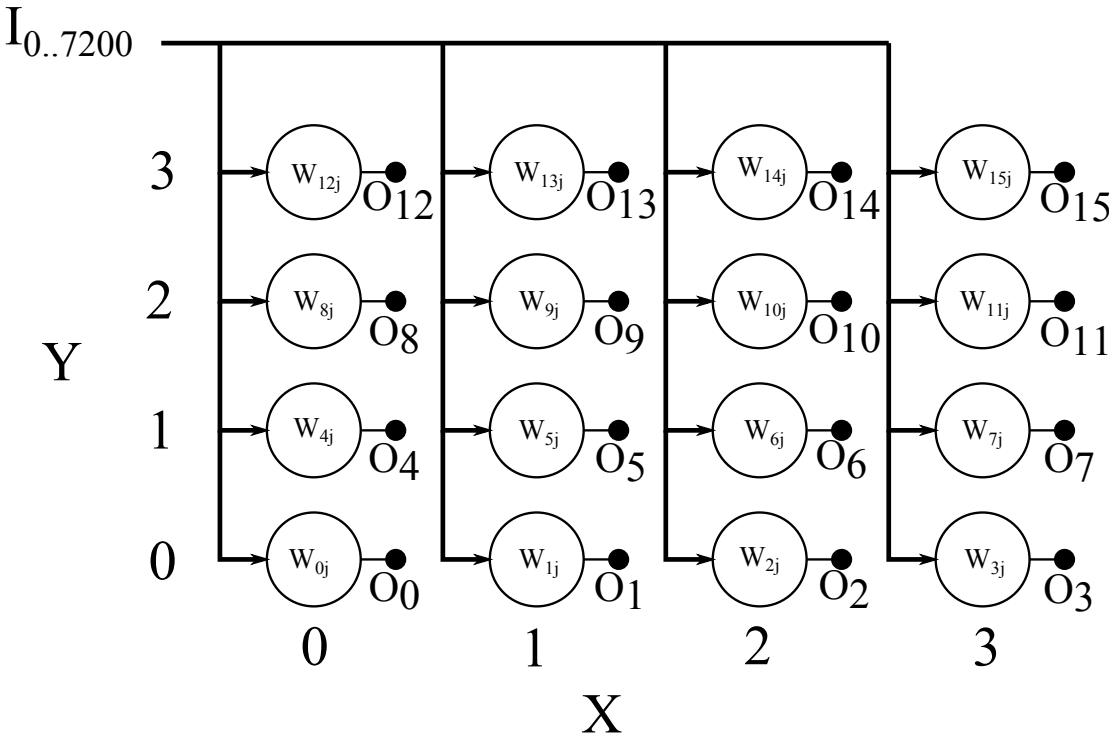


Figure 5.25: Kohonen 4x4 self organised feature map used to classify the Census 2011 variables.

The diagram in figure 5.25 shows the inputs and outputs for the network. The learning functions are defined as follows:

I_j	input j
O_i	output neuron i on grid
W_{ij}	weight j for output neuron i
d_i	distance from winning neuron to output neuron i
(x_w, y_w)	coordinates of winning neuron
(x_i, y_i)	coordinates of output neuron i
η	learning rate, clamped between 1 and 0.1
Λ	distance threshold, clamped between 4 and 0.5
e	training epoch number

Activation function:

$$O_i = \sum_j \|I_j - W_{ij}\| \tag{5.10}$$

Distance from the winner to all other output neurons (Euclidean):

$$d_i = \sqrt{(x_w - x_i)^2 + (y_w - y_i)^2} \tag{5.11}$$

Weights update function:

$$W_j = \begin{cases} W_j + \eta(I_j - W_j) & , d \leq \Lambda \\ W_j & , otherwise \end{cases} \tag{5.12}$$

Learning rate and distance function for training epoch:

$$\eta = \max(0.1, 1 - \frac{e+1}{10000}) \tag{5.13}$$

$$\Lambda = \max(0.5, 4 - \frac{e+1}{1000}) \tag{5.14}$$

The graph in figure 5.26 shows the network being trained to recognise all 2,558 Census variables. The mean error for all patterns is shown ('All' on the graph), along with the minimum error and maximum error for patterns in the training set. The network is trained until the changes to the weights become negligible. The final error value of 0.008 on the graph represents an error across the whole training set of 0.8%.

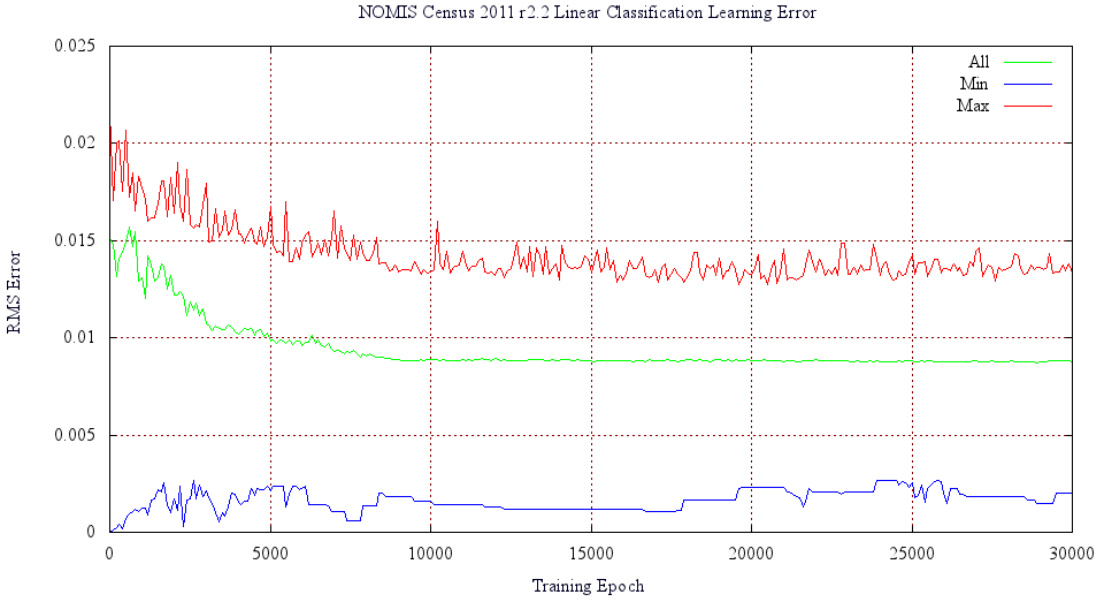


Figure 5.26: Maximum, minimum and average error for the Census 2011 data set during learning with a Kohonen linear classifier.

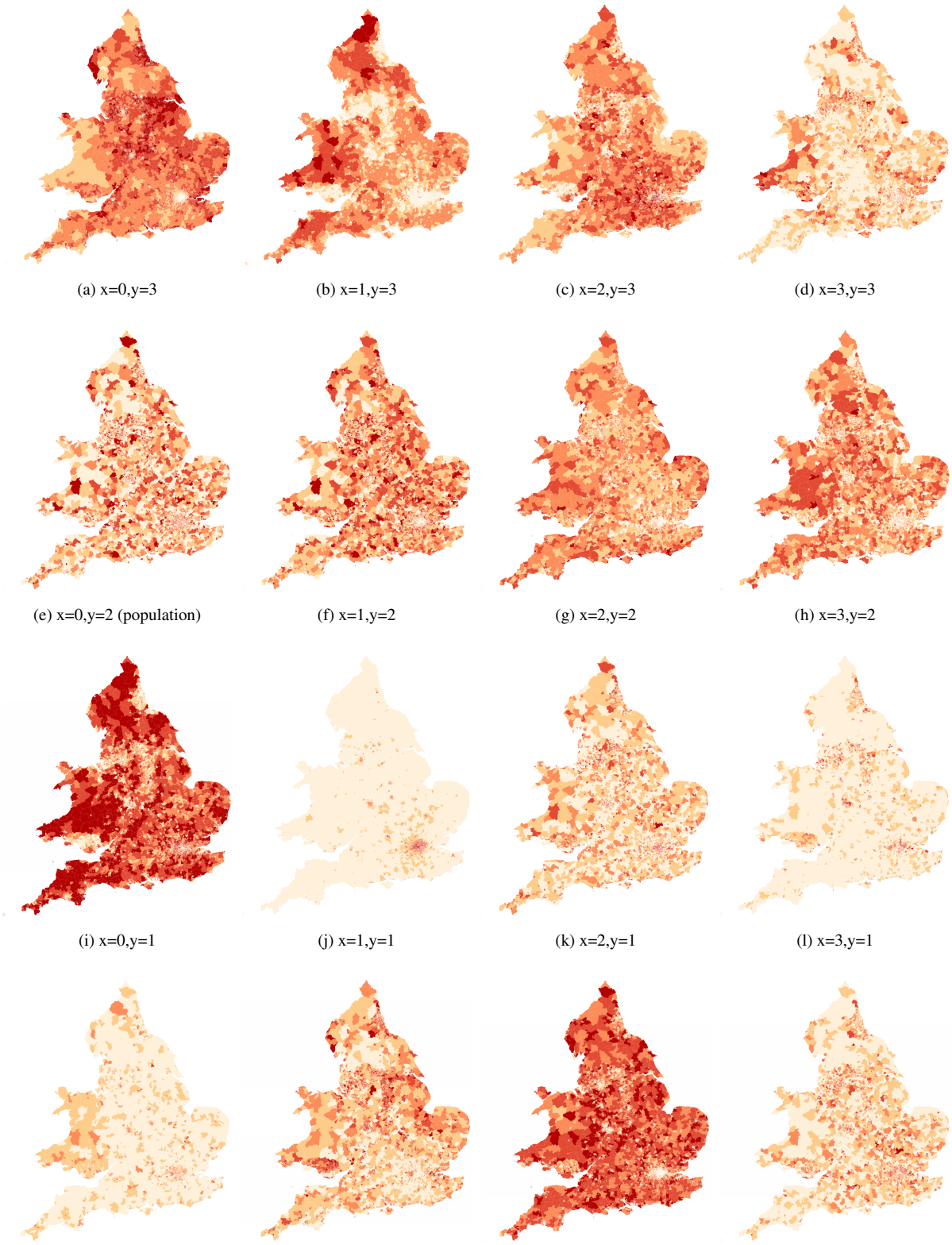
The reason for choosing this type of classifier is that the nature of the competitive learning algorithms means that the weights will converge on the cluster centres of the patterns that they represent. This allows maps to be plotted from the final weights to see geographically what the classifier has learned. The maps in figures 5.27a to 5.27p show these weights on an MSOA map of England and Wales with the (x=0,y=0) pattern in the bottom left and (x=3,y=3) in the top right. The maps have been drawn using Jenks breaks with 5 classes as the weights themselves are dimensionless.

After training the classifier, all of the training data can then be presented in order to obtain the classification for each and so start to form clusters. Table 5.7 shows the number of Census variables classified according to grid cell location ([0..3],[0..3]).

3	76	55	220	2
2	178	139	76	4
1	48	401	89	172
0	715	166	104	104
Y/X	0	1	2	3

Table 5.7: Number of Census 2011 variables classified to each grid cell.

From the non-uniform nature of the data it can be seen that this method is unlikely to be useful as a hashing function for maps. The highly inter-related nature of the Census data comes through in the higher number of classifications in the lower left triangle (1612) than the upper right triangle (529). The leading diagonal contains 408



(a) x=0,y=3 (b) x=1,y=3 (c) x=2,y=3 (d) x=3,y=3

(e) x=0,y=2 (population) (f) x=1,y=2 (g) x=2,y=2 (h) x=3,y=2

(i) x=0,y=1 (j) x=1,y=1 (k) x=2,y=1 (l) x=3,y=1

(m) x=0,y=0 (n) x=1,y=0 (o) x=2,y=0 (p) x=3,y=0

Figure 5.27: Weights maps.

classifications giving a total of $1612 + 529 + 408 = 2549$ for the full set of 2558 variables. The nine which are missing are due to variables in tables which have the same data value for every area and so are invalid. The top right cells, (3,3) and (3,2) contain the variables: *KS107EW0007* (male lone parent in employment), *KS107EW0016* (male lone parent total), *KS107EW0017* (percentage male lone parent part-time employment), *KS107EW0018* (percentage percentage male lone parent full-time employment), *QS607EW0043*, *QS608EW0043*, *QS609EW0043* and *QS610EW0043* (all semi-routine childcare occupations). These do all appear to be variables that are likely to be related to one another.

The next most important feature to be examined is the population count. By using the fact that the first variable in every table is the total count of population, the classification data can be scanned to see where these all appear. This is in cell (x=0,y=2), where 90 of the population count variables are located. The only tables not present here are the ones where the count is either for Wales only, or for communal establishments, shared housing or students (*KS405*, *QS419*, *QS402*, *QS421*, *QS603*). Finally, age structure from the table *KS102* is another interesting indicator. Looking at how the different age groupings either follow or diverge from the base population density shows some deviations. This table is present in the following locations: (0,0), (0,2), (1,1), (1,2), (2,0), (2,1), (2,2), (3,0) and (3,1). These represent most of the lower left half of the weights maps.

5.2.3 Conclusion

This section has introduced the idea of data mining sets of connected spatial data in Internet data stores and highlighted some of the problems. Making maps from all the data is a less computationally intensive task than spatial correlation as can be seen from the data flow diagrams in figures 5.15 and 5.16. The difference between the two is that the breaks algorithm used to make the map is of order $\mathcal{O}(k \times n \times log(n))$ [Hil15] while the spatial correlation is $\mathcal{O}(n^2)$. Then, additionally, each choropleth map is only rendered once as opposed to the spatial correlation which compares each unique pair of maps. While there are many more vertices making up the polygons for the map rendering than there are areas for correlation, the size of T in equation 5.3 grows fast enough that this form of naive data mining is computationally challenging. The opportunities to exploit parallelism at the algorithmic level and the potential for making new discoveries from the data highlight the need to develop this technique further, which is expanded upon in the following chapters.

How to cope with data at different scales is a question that has been temporarily avoided by the use of the Census 2011 dataset where all the data is available at the same geography. This is a critical issue which needs to be addressed as the aim of this technique is to make comparison of spatial data from different sources possible. It could be argued that this is part of the comparison, not the data identification and staging part of the tool, and so is the active part of the research. Many techniques for comparing two maps with different geographies exist and any investigation should try a number of different alternatives, so it comes back to the fact that it is the work flow that is important. If a viable work flow which makes investigation of data from disparate sources can be demonstrated, then that is a useful tool. It is about giving researchers the means to ask, and answer these types of question. Taking the data a step further, the work flow approach becomes even clearer when, in the next section, real-time flows of data are investigated.

Another question which remains is whether any persistent local storage is required, or whether the work flow can take the data directly from a data store and pass it to a vector tile renderer for visualisation. One technique is to convert everything to a single format, while an alternative technique is to store work flows which allow access to the data. Where data stores implement standards like Web Feature Service (WFS), this is just a case of protocols, but where data exists in formats like inside zip files or Excel

spreadsheets, then a more complex form of data staging is required. In the early days of the MapTube website, the London Profiler website used the GMapCreator tool to create all its London maps. These maps could be referenced directly by MapTube because of the compatible format. A similar type of framework is now proposed for how to handle the large Internet data stores, whether that be from *data.gov.uk*, the World Bank, or the U.S. site *data.gov*. Time constraints prevent the construction of another MapTube tool to handle the data staging and cleaning work flow, but is is easy to envisage how this could be achieved. The Data Store Miner developed in this chapter has been used to upload the Census 2011 maps automatically to the MapTube website which then allows the data to be used in other systems. An interesting open question is how far this type of process could be automated. Automatic spatial data identification and a web crawler process could potentially learn how to find new data on the Internet and visualise it directly. With a website like MapTube, where people can upload and build visualisations of maps, the machine could use these as training set examples when presented with a new piece of data. In an ideal world, though, the data would be left on the Internet and only the work flow used to obtain it would be stored on the MapTube site, rather than having to use a separate local source of storage which will always be limited by size.

To answer the question, "What does a data store look like?", which was posed at the start of the chapter, it looks something like figure 5.28, but it is the work flow used to reach this point which is more important than the graphic itself. By enabling different types of experimentation on data stores, the aim is to provide a new library of tools for researchers.

The diagram in figure 5.29 shows the tools and services presented in this chapter and how they relate to one another. The key concept is that the distinction between 2D, 3D and real-time visualisation has diminished. How these tools can be built into a system capable of handling real-time 3D visualisation is the subject of the next chapter.

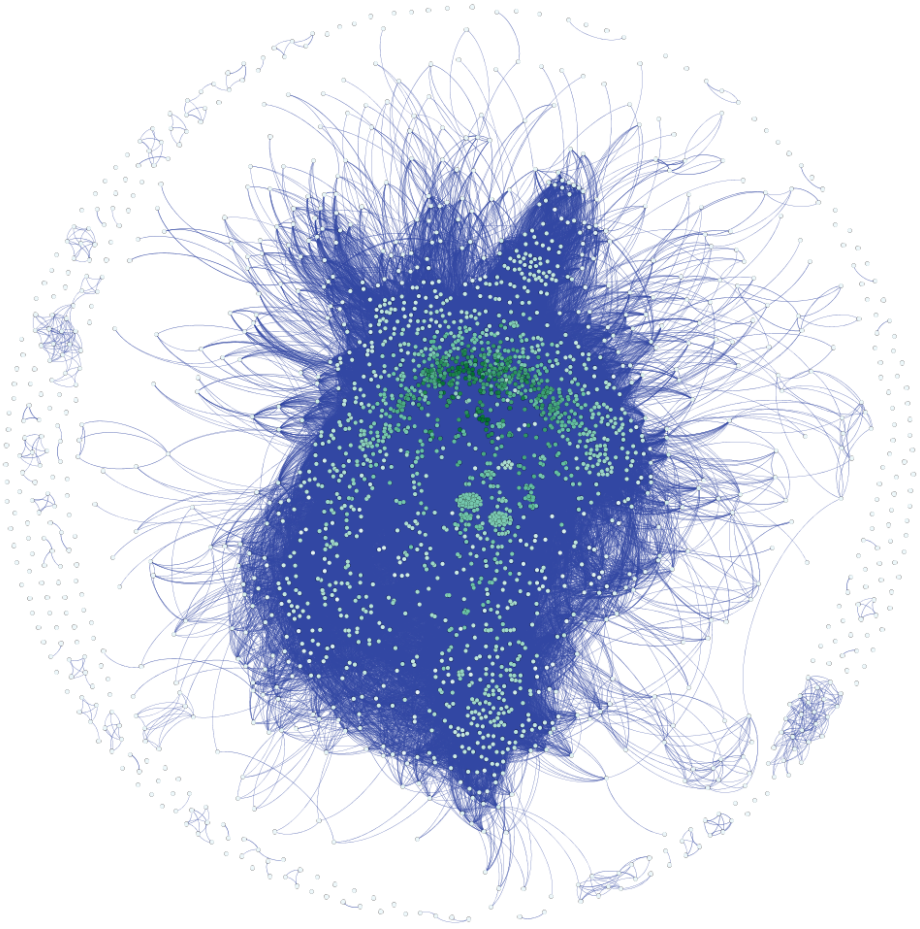


Figure 5.28: Network graph of the NOMIS Census r2.2 data release. The weights between nodes are derived from the spatial cross correlation of the 2,558 Census maps represented by each node. Nodes are coloured according to the number of connections. Even at this level of detail, distinct clusters are visible which suggest relationships. It is the ability to explore the relationships between the information in this way which is important.

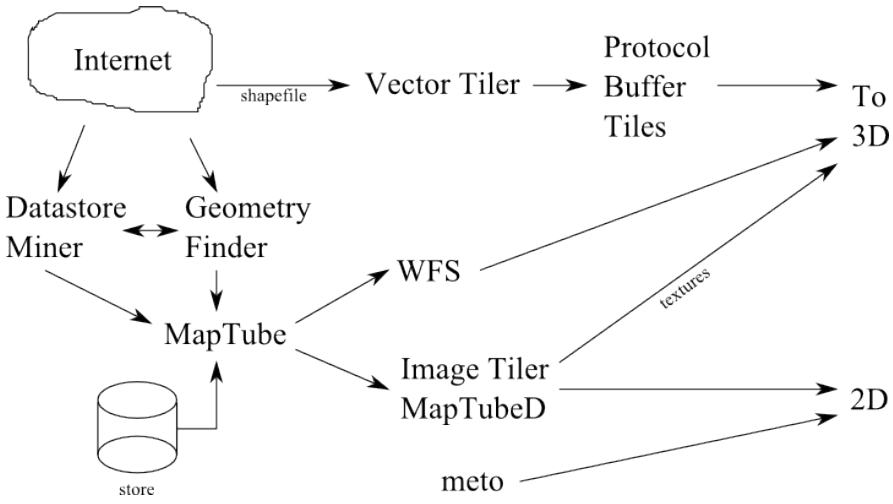


Figure 5.29: Overview of the services and software presented in this chapter.

Chapter 6

Real-time Mapping and Agent Simulations

While the previous chapter introduced work flows for handling geospatial data, this chapter extends the principle to real-time APIs. The visualisation also moves from 2D browser based to a higher performance 3D application due to the quantity and frequency of the data. Experiments outlined in previous chapters have explored browser based technology to its limits, so the next step is to see what can be achieved with a bespoke application designed around high performance desktop computing hardware. Everything in this chapter builds on the base created previously, with static data like roads, rivers and railway lines forming an integral part of the real-time simulation. This is where interoperability and geospatial standards are essential as a 3D simulation of city scale systems requires access to fabric and infrastructure data.

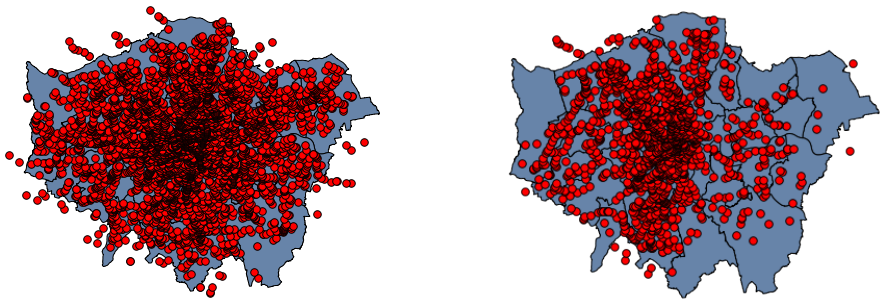
This chapter begins by looking at how to simulate the London Underground network using data from TfL's real-time API.

6.1 Example 3: Real-time Mapping and Agent Simulations

Real-time data is characterised by a work flow which provides access to data that is constantly changing and evolving over time. Data exists for multiple points in time, for example snapshots of the locations of moving vehicles, or sensor data for multiple fixed locations. Three sources of transport data are available for London with APIs providing real-time access to vehicle locations: "Network Rail Data Feeds" for National Rail trains [Net15], "Countdown" for TfL buses [Tra14a] and "Trackernet" for TfL London Underground tube trains [Tra14b].

Simply plotting vehicle positions, or animating them over time often shows a visualisation too complex to be useful. Figure 6.1b shows the effect of a bus strike in June 2012. This graphic was produced using QGIS with data extracted from the Countdown

API and transformed into a CSV file containing a snapshot of vehicle locations using a complex pre-processing work flow. In this instance, the API is a permanently connected stream, passing JSON formatted messages over *http* which contain estimated arrival times at every stop along the route. Whenever a bus moves in London and its estimated time of arrival changes, a message gets sent, so, at any point in time, the processing system will be tracking 120,000 live bus movement messages. The integration of this type of live data system with existing GIS systems is a challenge.



(a) Friday 29th June 2012, 9am. (b) Friday 22nd June 2012, 9am.

Figure 6.1: The bus strike took place on 22nd June 2012, with the data for the strike day at 9am and the week after at 9am plotted as a comparison. This figure is a version of the same data published in [Che+14].

> "Transport for London (TfL) revealed today that last week (week ending Saturday 13 December) was the busiest week on the Tube in its history. The London Underground (LU) network carried over 28.346 million people in that week. This exceeded the last record-breaking total reached in the week ending Saturday 13 August 2012 during the London 2012 Games when 28.235 million journeys were made."
>
> (Transport for London press release, December 2014[1])

Archives of data for trains, tubes and buses are available for the last 4 years, starting from July 2012 when London was hosting the 2012 Olympic Games. Given the pressure that was expected to be put on London's transport system during those months, the question being asked of all this data is this:

> "What type of tool is required to detect any unusual and interesting patterns in these streams of data?"

[1]Full version online at:
https://www.tfl.gov.uk/info-for/media/press-releases/2014/december/lu-breaks-olympic-record-for-weekly-passenger-numbers.

6.2 Adaptive Networks for complex Transport Systems (ANTS)

The 'Adaptive Networks for Complex Transport Systems' (ANTS) project is a library for handling London's real-time transport data relating to Underground, Network Rail, River Services and Bus vehicle positions and expected arrival times. Figure 6.2 shows the system diagram for the ANTS library.

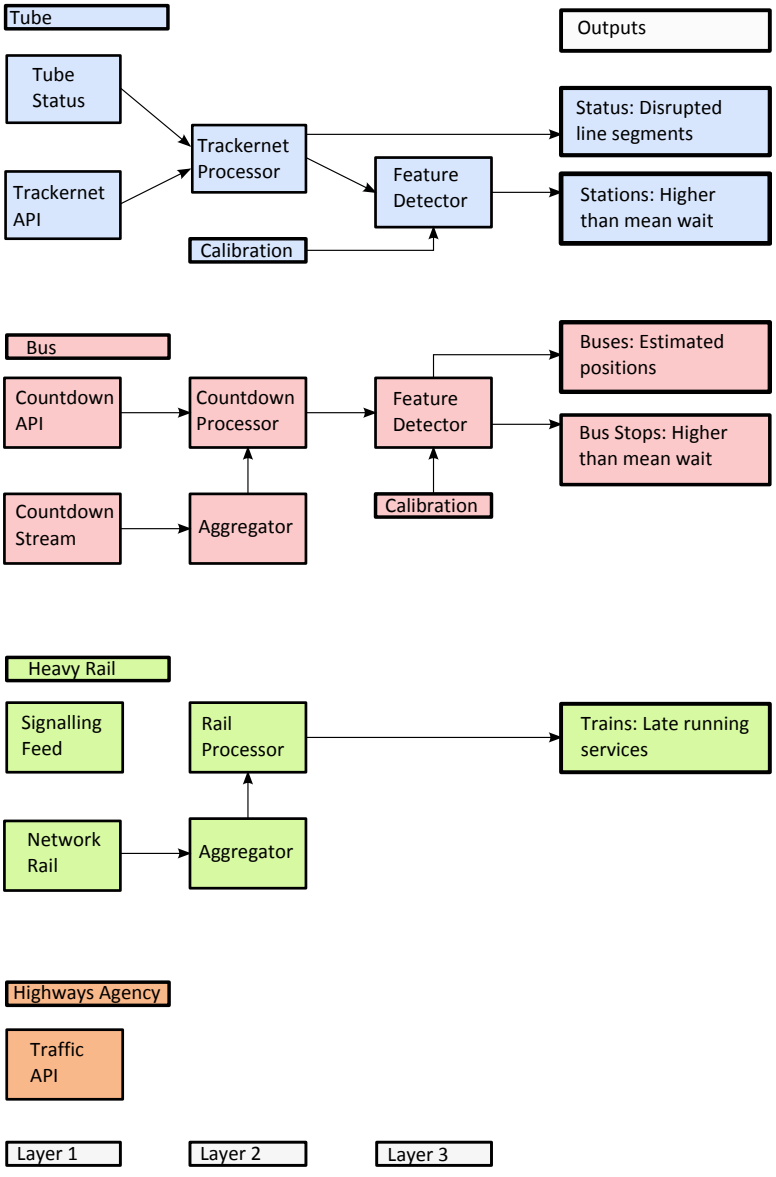


Figure 6.2: ANTS System Diagram.

This is a copy of the diagram that appeared in [Che+14], but with the addition of a 'Highways Agency' block after the traffic data was made available to the public in late 2014. This includes the state of all matrix and electronic roadside signs, along with traffic speed and journey time data for the whole of England and Wales. The ANTS library allows for the real-time tracking of 450 London Underground tubes, 900

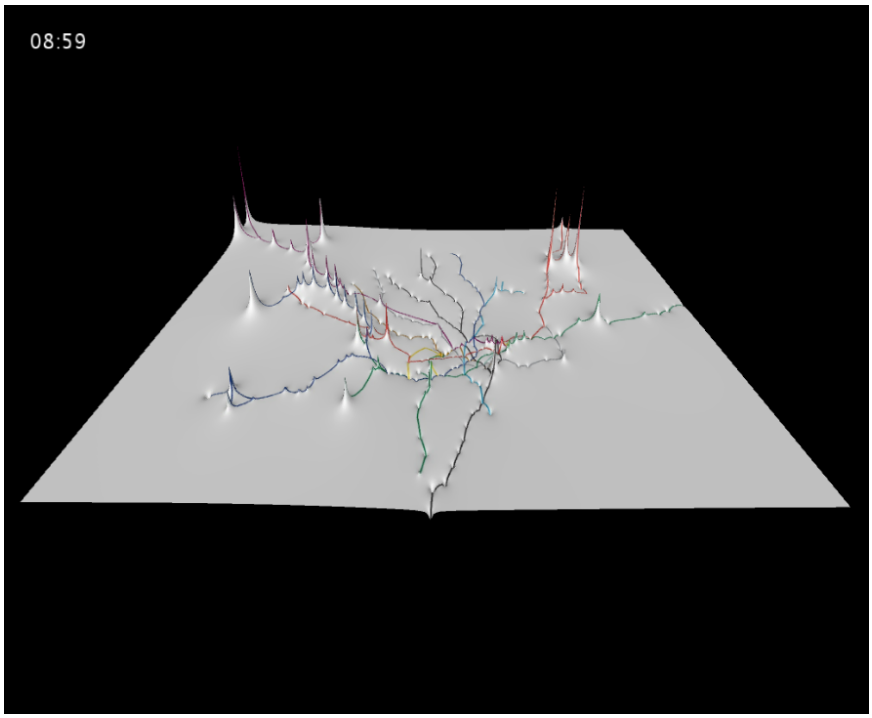
Network Rail trains, 7,000 TfL buses, 12 TfL river boats, 9,600 bikes and 17,960[2] traffic sensors.

The 'feature detector' elements in the ANTS system diagram use archive data for the bus and tube to calculate an expected wait time for every line, route, station, platform and stop for every hour of the day. Then any stations or bus stops with a waiting time one standard deviation above the mean can be flagged as delayed. Figure 6.3a shows a visualisation of actual waiting time built from the raw data which shows one of the problems with dealing with data of this complexity. When run for 24 hours, the waiting time, which is shown as the height of the delay surface, develops peaks suggesting delayed services when this is not the case in reality. The problem arises where multiple platforms are being used during rush hour, with services split between them. The solution is to measure the frequency of trains to specific destinations, which requires a deeper understanding of the data. Figure 6.3b shows another feature in the data, namely the number of buses and tubes running on each line, plus TfL's own service disruption data feed which is updated in real-time by a human line controller. In this figure, the gradual reduction in tubes to a skeleton service with the coloured bars showing disruption is due to the tube strike on the 28th April 2014. While this is an obvious example, the question is whether disruption can be detected or predicted from features in the data alone? The other point to note is that the Network Rail data already contains information on how late every train is against its timetable, highlighting the problems of fusing different types and sources of data.

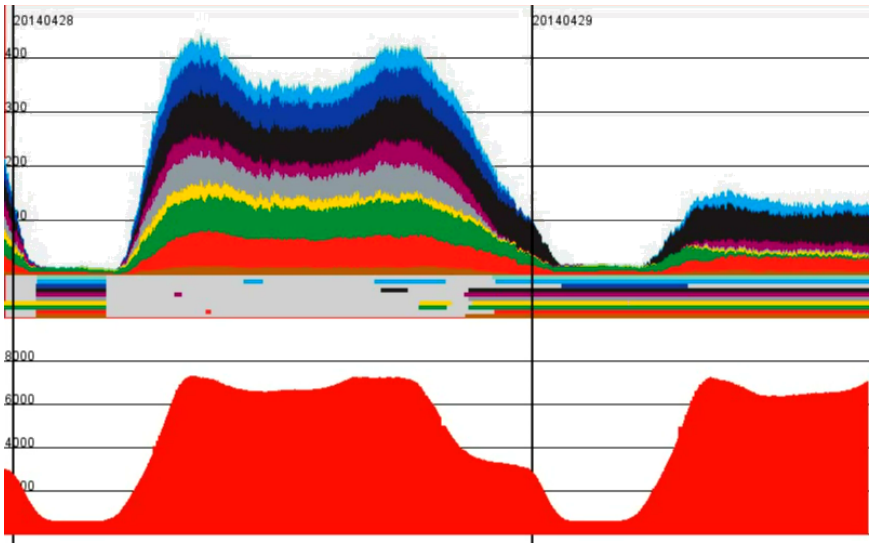
Drawing on the experience with the ANTS project and the real-time transport visualisations created for the ESRC Talisman project[3] and presented at the National Centre for Research Methods Festival in Oxford in 2014, there are two approaches to tackling real-time visualisation of vehicle positions. The data can be seen as a key-frame animation with linear interpolation between vehicle positions and times. Where this fails, though, is with real-time visualisation when the vehicle positions need to be forecast into the near future until new data is available in order to show continuous movement. Imagine the situation with live positions of London Underground tubes. If the API can only be queried every three minutes, then positions need to be forecast until new data is available and the view updated. By solving this problem using an agent based simulation, which uses rules to define the behaviour of tubes on the underground net-

[2] There are 6051 traffic concentration sensors, 6051 flow sensors, 4994 speed sensors and 864 travel time sensors.

[3] 'TALISMAN: Geospatial Data Analysis and Simulation' was an ESRC project (2011 to 2014), Reference: ES/I025634/1.



(a) ANTS Delay Surface.



(b) Tube and bus numbers for 28th April 2014 as a tube strike was about to take place.

Figure 6.3: The top figure (a) shows a delay surface created from the expected waiting time for a tube averaged over April 2014. Figure b shows the same data as an animated graph with the tube lines coloured according to their normal line colours as the top stacked histogram. Underneath are similarly coloured bars showing when disruption was reported on the lines. The bottom stacked histogram (red) shows the number of buses. The data shows the 28th and 29th of April 2014 as the tube network was shutting down for a strike on the 29th.

work, then the result can be used to display both real-time information and archived scenarios from previous situations. This leads to a very powerful model of a city which researchers can use to experiment on using real data.

This is the 'third generation', or 'programmable maps', application which can only be realised when agent-based simulations are integrated with spatial analytics. The approach taken here is to integrate the 'AgentScript' library into the MapTube website to provide programmable web mapping for 2D, while also developing a 3D GIS application, 'GeoGL', which implements the agent based modelling interface provided by 'NetLogo'[4]. The code for the 'GeoGL' project is open source and available at the following link: *https://github.com/maptube/GeoGL*. The results of these two approaches are analysed in detail in the next sections.

[4]The AgentScript library also follows the NetLogo example, although in Coffeescript.

6.3 Behaviours and Agent Based Models

Using the archives of London Underground, Bus and Network Rail data which have been built up over time, the following information can be extracted:

1. Vehicle numbers over the day, showing daily variation and peak usage of the system.
2. The routes vehicles take, where services are being created and where they terminate, along with frequency of service.
3. Locations of vehicles during the day, density, velocity and movement patterns.

The vehicles in question could easily be seen as agents in an agent based model, moving around on a road, rail or underground network. The rules can be derived following the ABM cycle proposed by William Rand in [Ran06] and shown modified here to handle online learning with real-time updates (figure 6.4).

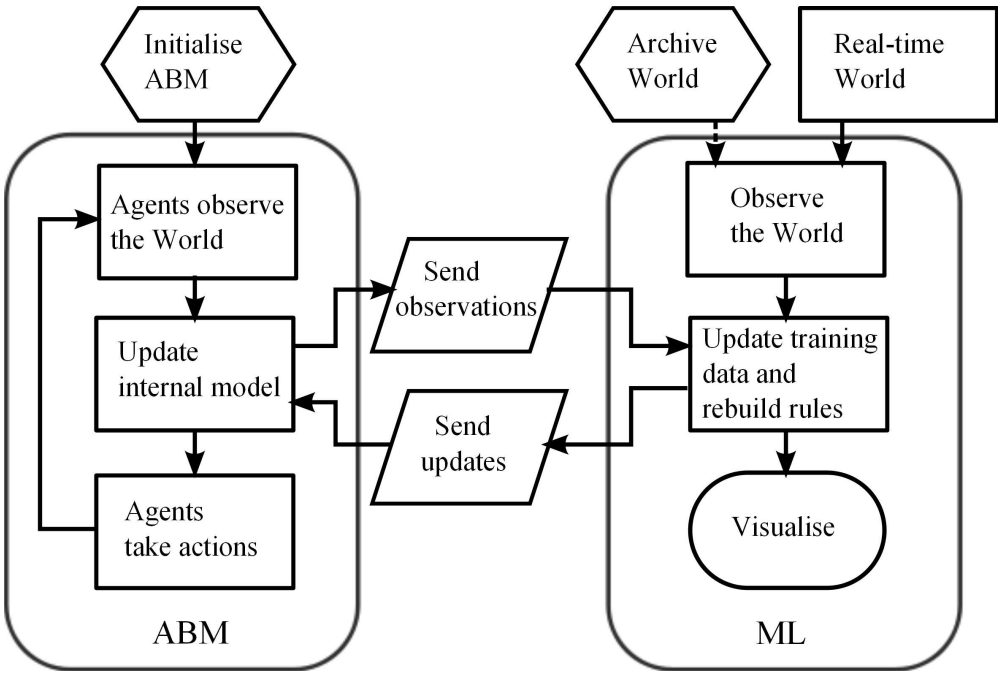


Figure 6.4: Agent Based Modelling (ABM) meets Machine Learning, based on Figure 3 from [Ran06].

This shows a closed loop cycle where information from the world is used to control the behaviours of agents contained within it. In the example of the London Underground tube network, agents can learn from the archive information how to mimic a normal day. This serves both as an accurate model which researchers can experiment on and also as a baseline for knowledge directed visualisation through comparison with the real-time information.

Algorithm 6 shows a simple tube behaviour model. This encapsulates the logic of tube trains on discrete lines moving between stations in a closed system. This is intended to be an "online" algorithm, receiving new information in real-time about tube positions which are used to modify the velocities via the "time to next station" information. Alternatively, in the absence of new data, a representative model of the tube network will continue running into the future. This is a model for analysis, not for presenting to users, so all information on data updates is available for analysis. The algorithm demonstrates the critical element in a library that could be used for real-time presentation of the data to and end user, but, here, there is the interesting situation of *all* the data being predicted. While the timestamps on the real-time data that this algorithm reads all show how much time has elapsed between the last data update and "now", every tube being positioned by this algorithm is predicted. There could briefly be a situation where a tube is described as "at platform" and displayed on the screen in the location where it is in the real world, but, one second later, it may well have left, or still be there. As the positions are all given to a level of accuracy and data updates are at fixed points in time, everything is essentially a prediction.

Algorithm 6 Tube Behaviour Model

Require: T, list of tube trains
Require: G, network graph of links weighted by link run time in seconds
Require: S, list of stations with geographical positions
for all Tube Lines L **do**
  for all Tube Trains in T on line L **do**
    Move train $Velocity/Time$ units along route
    if At Station **then**
      Determine the next station on the route (route choice)
      if Tube at end of route **then**
        Tube dies
      else
        Reset next station and velocity properties for this tube
      end if
    end if
    Update $Velocity$ based on latest time to next station and distance tube has to travel from G
  end for
  Check for new tubes created on this line (birth rate)
end for
return

By expressing the model in this form, the required inputs from the real-time data can be confined to the two decision points surrounding the route choice and birth rate. Everything else is defined either in terms of the network graph and the connections that exist between stations, or simple logic and velocity, distance and time. The model only

has to learn to predict when and where a route should start and what that route is, for example, “at 09:00 create a new Victoria line train at Brixton which goes all the way to Walthamstow Central”. Of course, this can not account for changes en-route due to operational reasons, but this is the type of anomaly which the program should be able to highlight in real-time mode.

In order to build a representative model of the tube network, the existing archives of running data are used to calibrate the system. Table 6.1 shows the real-time information from the ‘Trackernet’ API.

Variable	Description
Station Code	Three letter station code defined in appendix B
Station Name	Plain text common name of station used on information displays
Platform Name	Generally of the form giving the direction and a platform number as in “Northbound - Platform 1”
Platform Code	Either ‘0’ or ‘1’ giving the platform direction where North=0, South=1, West=0 and East=1
Train Set Number	Three digit numeric identifier for the train [000 . . . 999]
Train Trip Number	Two digit number counting how many trips this train has made [00 . . . 99]
Train Destination Code	Three digit number defining the train’s route and destination [000 . . . 999]
Train Time To Station	The number of minutes and seconds before the train reaches this station as “HH:MM”
Train Location	The train’s current position, for example, “Approaching Victoria”
Train Destination Name	Plain text of this train’s destination, for example, “Wimbledon”

Table 6.1: Information contained in the ‘Trackernet’ real-time API for the London Underground system.

The information available from this API is designed to be displayed on systems showing arrival times for tubes at a single specific station, so some pre-processing is required to extract individual vehicles and locations from the data. An added complication is that the data does not contain which line a tube belongs to and vehicle identifiers are not unique across all lines, so double counting of vehicles occurs if a naive approach is followed. However, the destination codes can be used to uniquely identify the line, but these codes are not published by TfL. With the quantity of data available, this code table can be derived using a simple data mining technique. By filtering the data for destination codes used at stations served by only one line, the table can be built up over time. The only exception to this is the Waterloo and City line, which, having only two stations at Waterloo and Bank, is not unique. Due to the special 5 train service that operates between these two stations, these codes can be extracted through their frequency of service and exclusive use at these two locations. Building on this technique,

by storing every destination code and where it is used, a simple logical elimination can discover additional codes. For example, if code '88' is used at Euston (Victoria and Northern lines), and is then seen again at Green Park (Victoria and Jubilee lines), then it must be a Victoria line code by a process of elimination. Once this information is extracted into a code table, other useful analyses can be performed on the data.

The following relation expresses the data extracted from the API:

$$I = \{Station, LineCode, DestinationCode, TripNumber, SetNumber, ArrivalTime\}$$

The $ArrivalTime$ is derived from the 'time to station' and the data time. The data time is based on 480 equal 3 minute time slots spaced throughout the day, which is the frequency at which the 'Trackernet' API can be queried for new data. All the data archives contain data for each of these 480 time points for every day of the year. After initialising the data table, I, with all the data for January 2014, amounting to 173,515 records, the critical factors which govern the operation of the network can be derived. This is where the need for simulation software which can handle this level and complexity of data is realised. The following sections go through the process of deriving the critical information required to build the model of the tube network, while at the same time ensuring that the software is generalised to allow for the bus network and sensor based data such as air quality and weather which are linked to mass transport systems. Initially, the data processing required to learn the model behaviour is covered, then in the later sections, this is coupled with real time visualisation. This is a model for analysis, built to prove that the tube behaviour can be learnt from the data available, so the calibration has been through the 2014 dataset mentioned earlier. There is no reason why a different set of data could not be used, even from another country's metro system, but the 2014 data set was the most complete at the time of writing this thesis and is sufficient to provide the data analysis for this section and the "Data Exploration" section later. While some of the algorithms presented later in the section can be run in an "online" mode, this is not currently being run. However, in the event of a real live project requiring online real-time learning of data directly from the stream, then a suitable application can be built based on the theory presented in this chapter.

At this point, only the ABM part of the 'GeoGL' program is used to derive statistics from the block of 2014 data using the ABM classes: 'Model', 'Agents', 'Agent' with basic loading of the data archived by the ANTS library implemented with a custom

comma separated (CSV) reader utility. Graph processing and ABM 'Links' are brought in later when route choices are being derived. In the initial stages, only the ability to read the data and perform simple functions which count numbers of agents based on specific properties are required.

The final results of the derived destinations code table are listed in appendix C. This detects 180 unique route codes in use during January, from a total of 251 codes in use, leaving 71 codes unknown. Using this data, the total number of unmatched destination codes equals $1,843,964$ against the total of $38,386,381$ matched codes, resulting in a match of 95.4% of the data. The derived destination code data is shown graphically in figure 6.5 with the number of times each of the top 100 codes was used. As can be seen from the graph and from the data, the Hammersmith and City and Circle line routes, along with the Waterloo and City line are more difficult to identify automatically due to the lack of unique stations serving only one line. The only unique stations on the Hammersmith and City and Circle lines are along the Hammersmith to Paddington section. The rest of the routes are shared with other lines, so extracting the information requires the more complex logical reduction outlined earlier. Similarly, the Waterloo and City line only runs between Bank and Waterloo, both of which are shared stations.

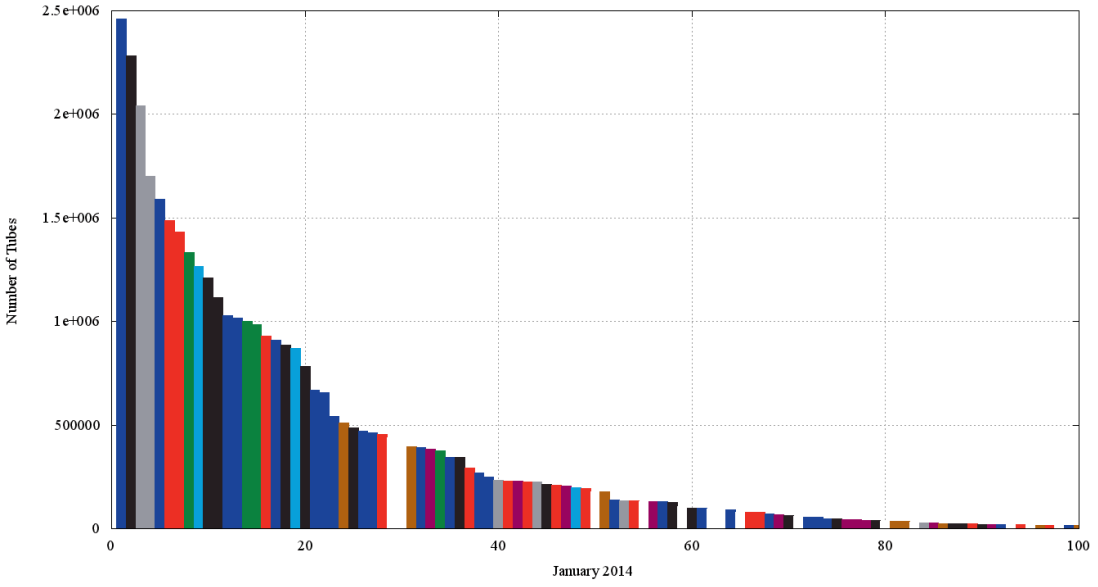


Figure 6.5: January 2014 weekday total destination codes used by line and frequency. Only the top 100 destination codes out of 251 have been plotted. The X axis shows the TfL destination code number, while the Y axis plots the usage count.

Further investigation of the data reveals some additional inconsistencies, for example, 'Check Front of Train' before the route has been assigned while a tube is moving

from a depot to its starting position. As with any real world data, analysis shows up inconsistencies where destination codes have either been used in error, or have become corrupted in transmission. Interestingly, there appear to be consistent places on the network where wrong information appears, suggesting it is a feature of how the TfL systems are programmed. As an example, a Victoria line train at Euston with a destination of Brixton always shows up as a northbound train when in fact it should be southbound.

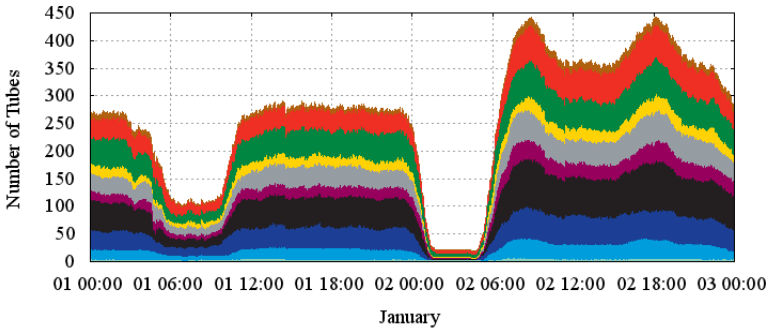
This solves the line identification problem, so now the number of unique trains running on each line can be determined by joining $\{DestinationCode, Line\}$ with the original dataset, I. Figure 6.6 shows the counts of the number of trains running on each line for the whole of 2014. Normal weekday running is evident, with a distinctive morning and evening rush hour peak. During 2014, there were three failures of the API on 13 February, 10 April and 18 August. The first two were caused by a software update on the server at 3am not completing cleanly with the server stuck at the reboot stage. The August failure appears to be an API failure with the ANTS software not reconnecting to the TfL API successfully after the outage was cleared. Days with lost data are shown highlighted in yellow, which raises an interesting question with regard to all data derived from an API. Given an unusual data point, how can we be sure that this is a true representation of real life and not a failure of the algorithm or hardware? Ultimately, the data is nothing more than an imperfect window on to the real world, so the only recourse is to decide whether the outlier point could fit into the body of existing data, or whether it is so unusual that it has a higher probability of being erroneous. The key lies in the supporting evidence from other, independent sources.

Figure 6.6: Tube Counts for 2014. Data collection failures are shown as yellow highlights.

Following on from the plot of all the 2014 data, other notable days in the dataset are:

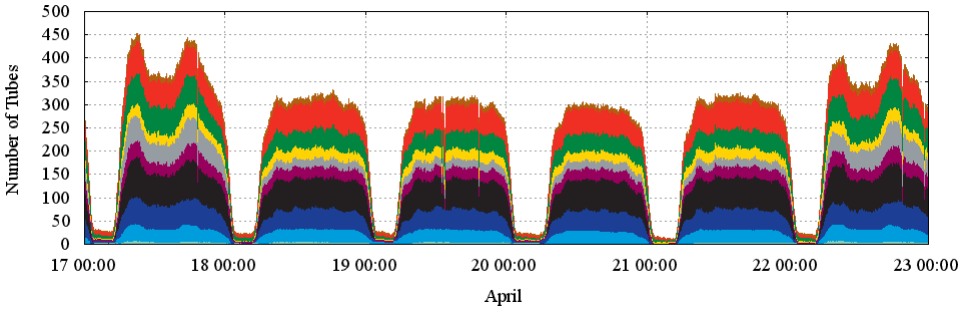
January 1

Note tubes running overnight (01 Jan 00:00), the Bank Holiday (Wednesday) and the first working day (Thursday 02 Jan 06:00) with a morning and evening rush hour.



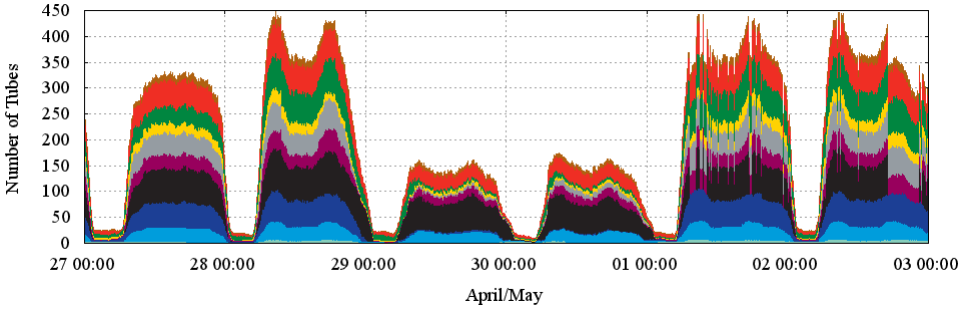
April 18 to 21

Easter Bank Holiday, Friday to Monday four day weekend. The chart starts on the preceding Thursday and finishes on the first day back at work, Tuesday 22nd.



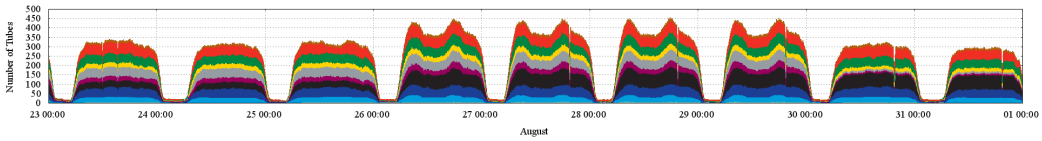
April 28 to May 2

Tube Strike, limited service on most lines. Sunday April 27th to Friday May 3rd is shown. Problems with the service on the days after the strike are evident.



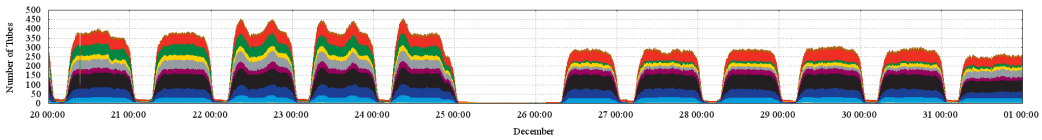
August 25

August Bank Holiday Monday, 3 day weekend, 4 day week.

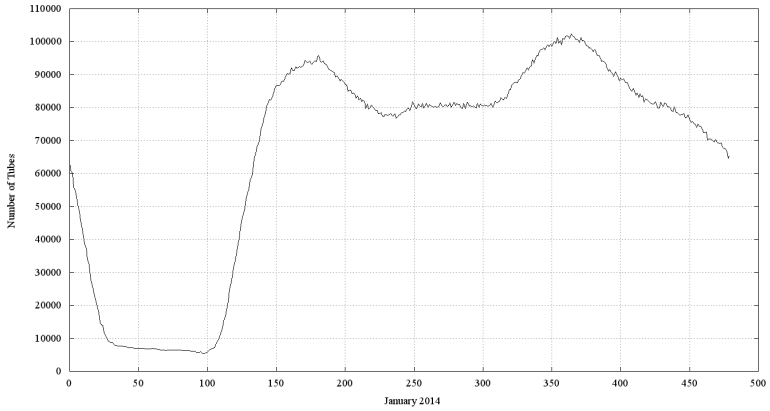


December 25

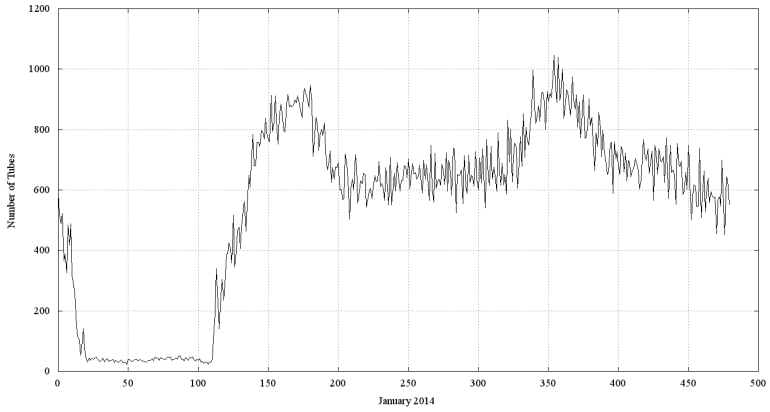
Christmas Day until January. The only day of the year with no tubes.



By combining all the data for weekdays during January 2014, an idealised weekday variation graph of tube numbers can be plotted. This information is shown in figure 6.7a, along with the same information, but for only the tubes going through Waterloo, in figure 6.7b.



(a) All tubes.



(b) Waterloo only.

Figure 6.7: January 2014, weekday total number of tubes and also for Waterloo only.

The final requirement before a model can be created is to analyse the routes, route choices, frequencies of service and creation of new tube services. This amounts to the derivation of the timetable from the real-time information stream, which is a data mining task. While the numbers of tubes running on each line can be estimated based on an idealised day, whether weekday or weekend, tubes must be created in the correct locations and travel along a prescribed route to their destination. In order to build an accurate model, this information needs to be extracted from the real-time data. A primary key for vehicles is $PK_{tube} = \{Day, DestinationCode, Trip, Set\}$. Alternatively, the $Line$ can be used in place of the $DestinationCode$ as there is a unique mapping between the two sets. The first task is to identify where new services start from, which can either be achieved by looking for the end points of routes, or by finding the first occurrence of each PK_{tube} key every day.

The frequency and location of where services are being created can be extracted by finding all the unique services running each day. The unique tube trains running on each day can be expressed as the following relation:

$$T_{unique} = DISTINCT(\{Day, DestinationCode, Trip, Set\}) \tag{6.1}$$

The keyword *DISTINCT* is used here as defined in the 'SQL-92' reference, selecting every combination of tuple that exists in its argument once only.

Then the creation point is the minimum time associated with a service, while the maximum time identifies the end point of the service. When aggregated by time, this gives the creation and destruction rates:

$$T_{new} = \overset{min(\{TimePoint\})}{I(u)} \quad \forall u \in T_{unique} \tag{6.2}$$

$$T_{end} = \overset{max(\{TimePoint\})}{I(u)} \quad \forall u \in T_{unique} \tag{6.3}$$

As the purpose of this chapter is to create re-usable tools which can be used for this type of work, the modelling framework described previously has been used to load and parse the files from the ANTS archive for January 2014. This code is then extended to find the T_{new}, T_{end} points along with the grouping by $DestinationCode$ which allows the geographic routes to be formed. It is worth nothing that, along with the vehicle creation data, it is also possible to build the network graph of the tube

system directly from the times and running information as the order that vehicles visit the stations implies the physical links between them. This methodology also forms the basis of Google's GTFS (General Transit File Specification) format, being derived from 'passing points'. These are fragments of information which say that a particular vehicle passed (and possibly stopped) at a particular station on a network at a point in time. Using this information it is possible to piece together timetables and routes. The code being designed here takes this a step further, though, by proposing that the passing points have probabilities which are used to create a stochastic model based on real-time data which is then used in the simulation. Using this method, a real-time event can be labelled as common or unusual based on the baseline archive data. This technique can be applied to the bus network and the train network, so is useful as a general transport system library. Automatic construction of the London Underground network graph proceeds by tracking the tube agents as the real-time data causes their 'next destination station' to change. By tracking the previous station and the next station when it is changed by new data, an origin destination table can be built up over time. This needs to be recognised as a probabilistic network as, with a 3 minute sampling period and 1 or 2 minute runlinks between geographically close stations, some stations can be missed. This type of real-time tracking is useful for gauging reliability of service, though, and is used in parallel with the analysis of archive data.

The Agent Based Modelling layer in the system has been implemented using the same design pattern as the NetLogo system of [Wil99] and the AgentScript implementation in Javascript. One reason for this is simplicity, but, by using related implementations of the same underlying structure, both the MapTube website and the GeoGL application can use variants of the same models with very minor changes. The decision also permits the re-use of models from the NetLogo community with the same minor changes. Figure 6.8 shows the 'Model' class which controls the 'Agents' and 'Links' required for the London Underground simulation. No concept of 'Patches' is used in this first implementation and one further deviation from the NetLogo world worth mentioning is the graph and network structure. The links (track) between two agents (stations) define the structure of the network, but a 'Graph' class which contains the 'Edges' and 'Vertices' of the network is maintained. In order to build a visual representation of the graph in the virtual world, the 'NetGraphGeometry' class is used to build the tube geometry from the network graph's edges using the 'Tangent, Normal,

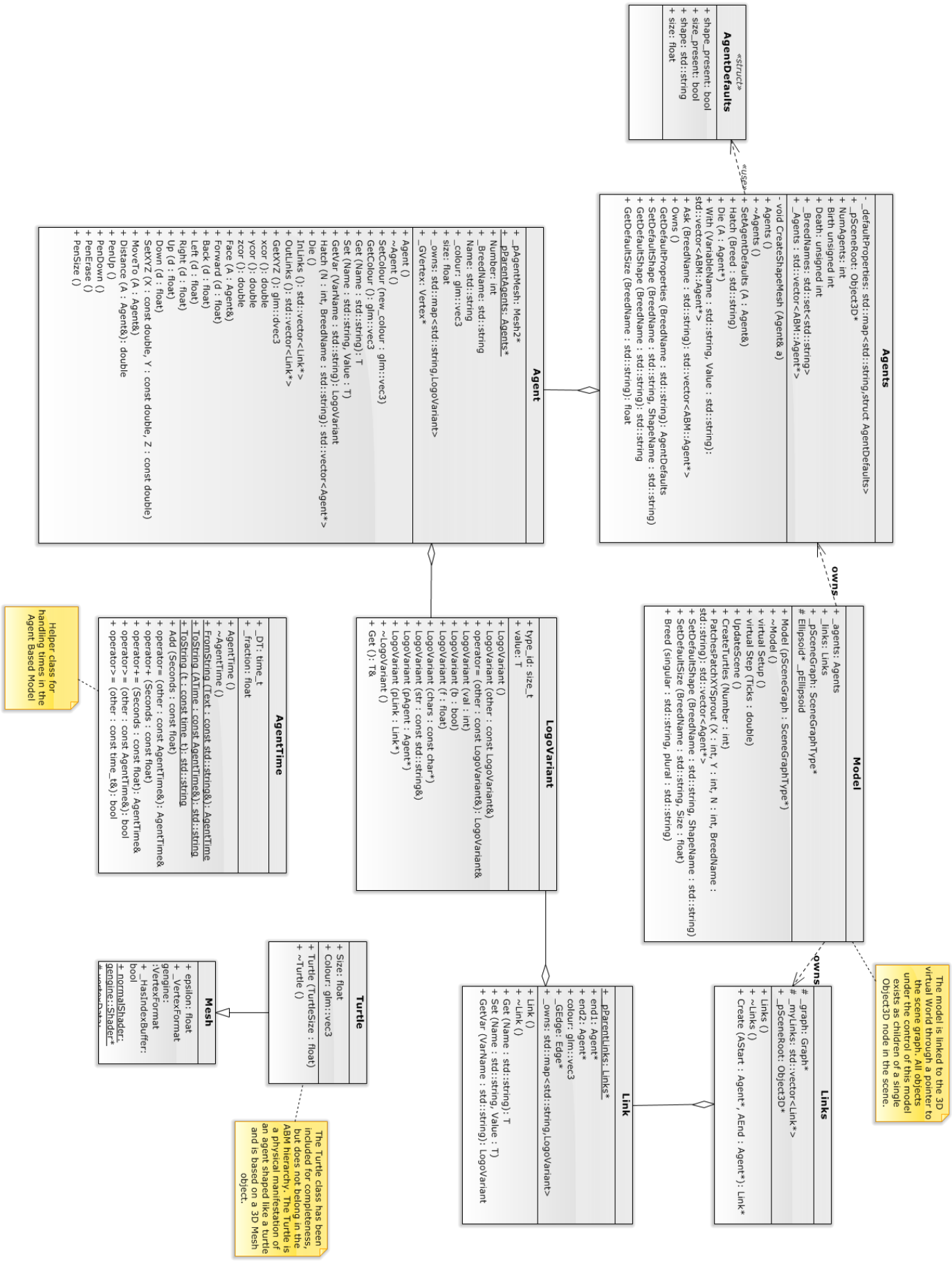


Figure 6.8: GeoGL ABM class diagram.

Binormal' (TNB) method. All agents and links are added to the scene graph under a common ancestor (Object3D class), with minimal linkage between the ABM layer and the Virtual Globe layer. Agents are required to keep a pointer to their object in the scene graph for when they move, change colour or change shape, but otherwise the two systems are completely separate. The ABM layer has the task of keeping the physical manifestation of the model up to date in the scene and nothing more. No communication between the scene and the ABM takes place. More information on the rendering system is given later in the chapter, with a class diagram in figure 6.25 showing the 'NetGraphGeometry' and 'Object3D' classes used to render the agents.

The loading of agent data from archive files and real-time APIs is accomplished using a C++ utility class with a lambda function run on every line of data to handle the construction of new agents from the fields in each CSV row. With loading, creation and destruction of agents, instrumentation is now possible, even without a fully animated model running. This level of basic functionality is enough to be able to count numbers of tubes running on the individual lines and extract information like destination code usage and route choices. Data-mining statistics like speed and headway are possible from the data contained in the agent set. This is a form of stream mining, where the constantly changing stream of input data is used for feature detection. Route information can be extracted by storing origin destination data when tube agents choose their next destination. This also gives probabilities for routes which can then be used in the analytics later.

The data in figure 6.9 shows the rate at which tubes are being created and destroyed, smoothed using a four sample moving average (12 minutes). Figure 6.10 shows the graph of creation minus destruction. By calculating a cumulative total of the area under the service 'creation minus destruction' graph, the average daily variation shown in figure 6.7a is approximated. This is used as an additional verification method.

Problems occur because the London Underground never shuts down completely, so, by forcing the time window to be between midnight and midnight, all the services running through until 1am are terminated prematurely, giving the large peaks seen at the left and right hand edges. Despite this, the information extracted here can be used to determine the frequencies of services when the analysis is widened to a per station basis.

One further point to note is that, due to the sampling interval, neither T_{new} nor T_{end}

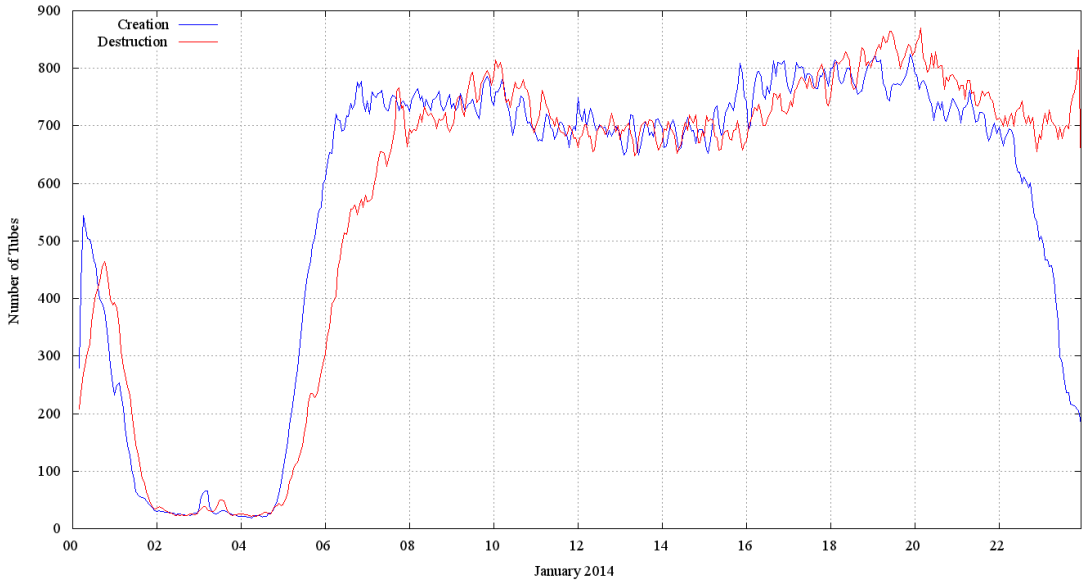


Figure 6.9: January 2014, average numbers of tubes created over all weekdays. Smoothed using a four sample moving average filter. The X axis shows the hour of day.

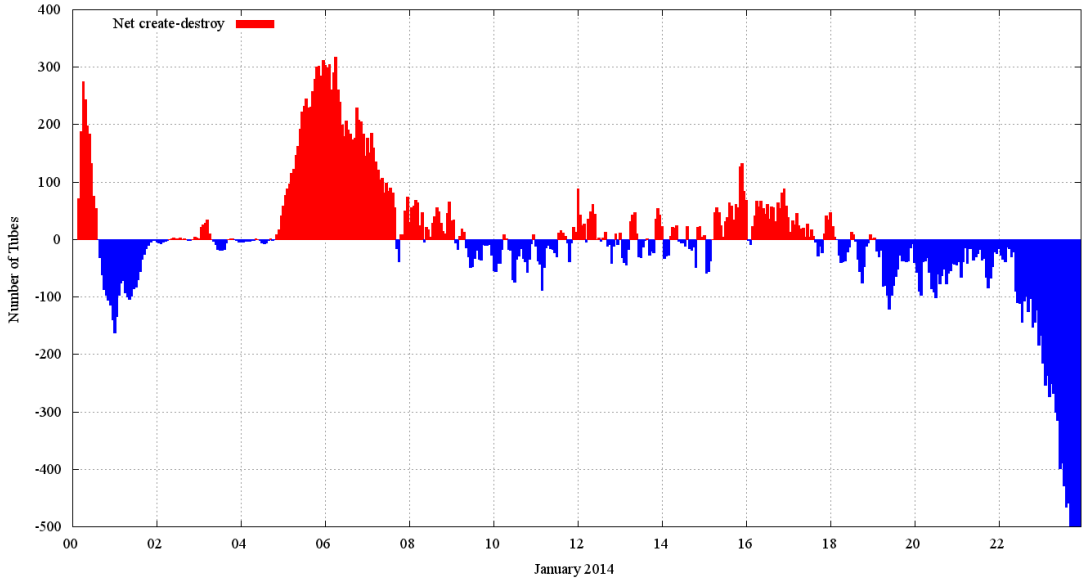


Figure 6.10: January 2014, net tube numbers (creation-destruction) over all weekdays. Smoothed using a four sample moving average filter. The X axis shows the hour of day.

will necessarily occur at the stations on the start and end of the full route, and, to further complicate the problem, the destination codes on the tubes may not get changed until the service departs. Despite this, the time error is bounded by the three minute sampling rate, so this suffices as an indicator of the rate at which new services are being created and existing services end.

Using this information about service creation, it is now possible to answer the 'birth rate' question from the decision point in algorithm 6. It is not possible to derive an accurate tube timetable from this real-time data due to the sampling rate of 3 minutes which is greater than the 1 or 2 minute frequency of service seen during the rush hour, leading to aliasing. However, it is possible to work out probabilities for where new services are being created throughout the day, leading to a method of analysis which can be used to determine how far the current system is from 'normal' running. While the sampling rate is only 3 minutes, the information contains tube arrival and departure times for every stop along the route, allowing the calculation of timing points at greater than the 3 minute sampling interval. The data for January 2014 shows the times and stations where new services are being created as a subset of all stations, S. While new services can, theoretically, be created anywhere on the network, the points at the ends of lines and where there is a change in frequency are the ones that are timetabled. These show a higher probability of service creation than the other stations. Any discontinuity in frequency along a route means tubes have either been inserted or removed.

Starting point stations are defined as the subset of stations where new services (T_{new}) start from, using the tube identifier of $\{DestinationCode, SetNumber, RunNumber\}$, along with the $\{TimePoint\}$ and $\{Day\}$ as a primary key into the original information:

$$PK_{service} = \{DestinationCode, SetNumber, RunNumber, Day, TimePoint\}$$

$$S_{start} = \{I(DetailsStationCode), I(PK_{Tube}) \in T_{new}\} \tag{6.4}$$

Equation 6.4 selects the $DetailsStationCode$ field from the original data table, I, at the points every day when new services start from (T_{new}). This results in 261 stations out of a possible 263 where new service creation is detected. By counting the number

of occurrences of each distinct station in S_{start}, the distribution in figure 6.11 can be obtained. This shows the counts for how many services start from each of these stations.

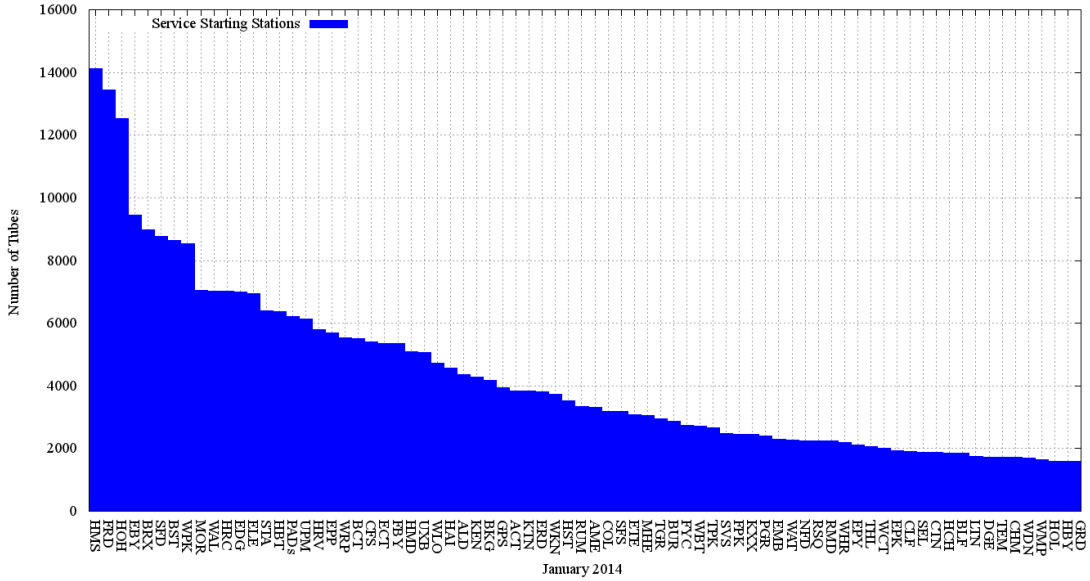


Figure 6.11: January 2014, stations where new services are created over all weekdays. The counts of number of services created are plotted on the Y axis and sorted from highest to lowest. Only the top 75 are plotted here. The three letter station codes are listed in Appendix B.

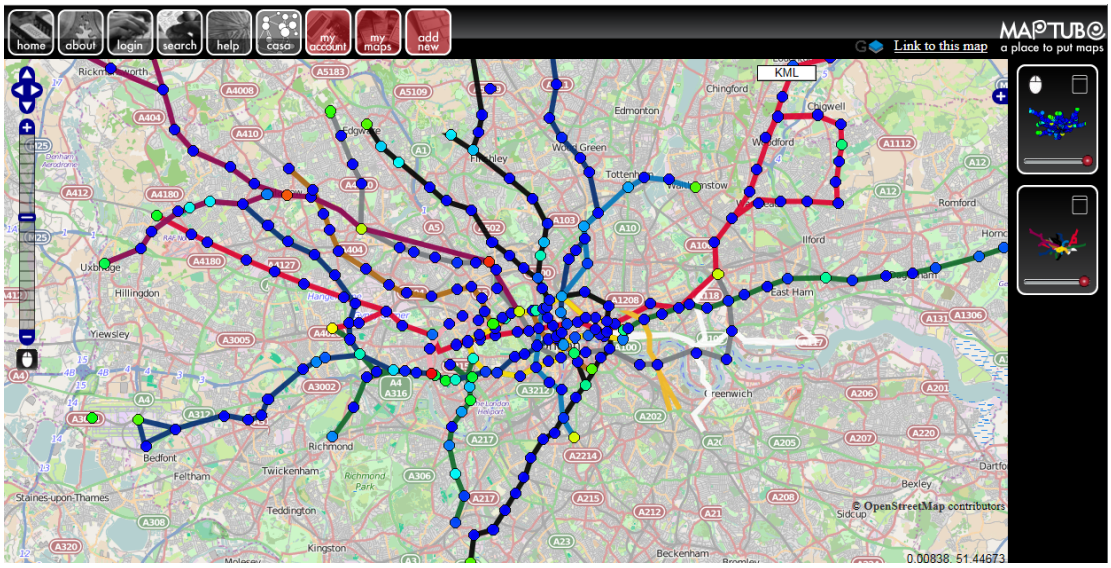


Figure 6.12: January 2014, probability of new service creation at stations over all weekdays, plotted using MapTube. This map is available online at the following address: *http://www.maptube.org/map.aspx?mapid=1183*.

Based on there being 31 days times 480 timepoints per day in January, this gives a total of 14,880 timepoints for the month, allowing a probability to be calculated for the data plotted in figure 6.11. The top station is Hammersmith (HMD) on the District and Piccadilly lines with $P_{new}(HMD) = 14,114/14,880 = 0.95$. This is usually

a result of the same station code being used on two lines, but picking out Brixton (BRX) on only the Victoria line, $P_{new}(BRX) = 0.6$. The map in figure 6.12 shows the probability of new service creation for all the stations on the London Underground. This shows visually where the creation points are on the network and where service frequency changes.

Having identified a route creation point, taking one of these stations which serves a single line as an example, the data for Brixton (BRX) shows the times when new services are likely throughout the day. The destination codes in use at Brixton, on the southern end of the Victoria line can be determined from $I(\{DetailsStationCode = BRX\})$, as shown in table 6.2.

Destination Code	Destination
88	Brixton
341, 342, 397	Seven Sisters
458	Walthamstow Central
459	Warren Street
514	Victoria

Table 6.2: Destination codes in use on the Victoria Line at Brixton during January 2014.

The graph in figure 6.13 shows the arrival times during the day of tubes between Brixton and Kings Cross as an aggregate of all weekdays in January 2014. As this is based on arrival times, the data is at a finer temporal scale than the 3 minute sampling interval.

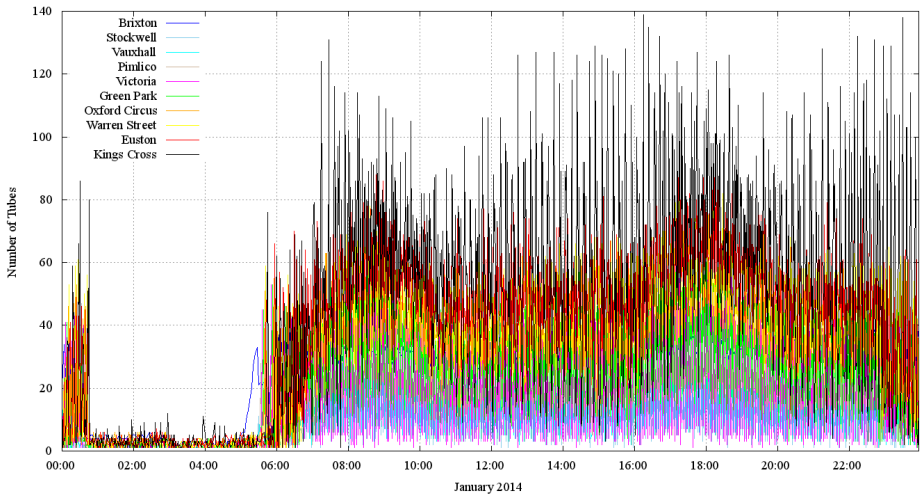


Figure 6.13: Times and number of trains detected as being at Victoria line stations between Brixton and Kings Cross as an aggregate of all weekdays in January 2014.

Data for Brixton, where the route starts from, can be seen as falling on the sampling points, as the departure time isn't specified in the data, while at further points along the

route, the expected arrival time is at a 1 minute resolution. More tubes scheduled to arrive at a station can be detected further along the route, which explains the vertical stacking of the graph. Figure 6.14 shows the same data, but for a segment of time between 8am and 10am.

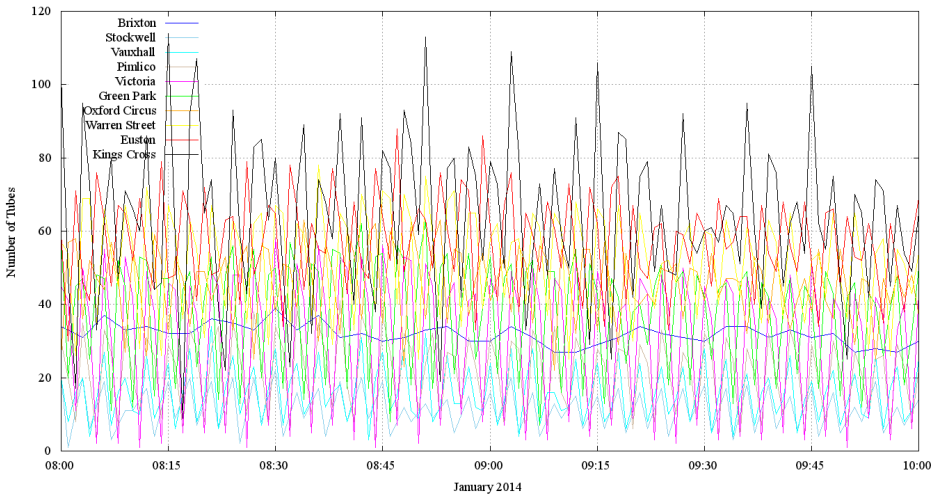


Figure 6.14: Times and number of trains detected as being at Victoria line stations between Brixton and Kings Cross as an aggregate of all weekdays in January 2014. The plot shows the segment of time between 8am and 10am.

From this data it is possible to obtain the approximate arrival times which are required for the model. Based on this data, the probability of seeing a tube at any location during the day can be calculated, which allows for comparison with the real-time data to detect any unusual events. By including an offset based on the amount of time it takes a tube to get from one station to another[5], the modified graph in figure 6.15 is a first attempt at deriving a timetable from the 1 minute running data. Just for completeness, figure 6.16 shows only the Kings Cross data for a comparison.

[5]Between Brixton and Kings Cross, all the times between stations are 120 seconds, apart from Warren Street to Euston which is 90 seconds.

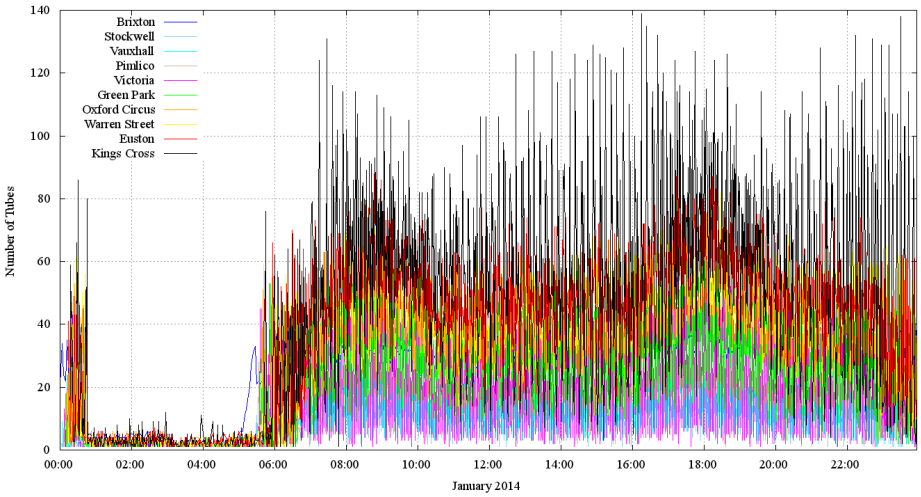


Figure 6.15: Times and number of trains detected as being at Victoria line stations between Brixton and Kings Cross as an aggregate of all weekdays in January 2014. Times for stations after Brixton are offset to reflect the time between stations to show the timetable points.

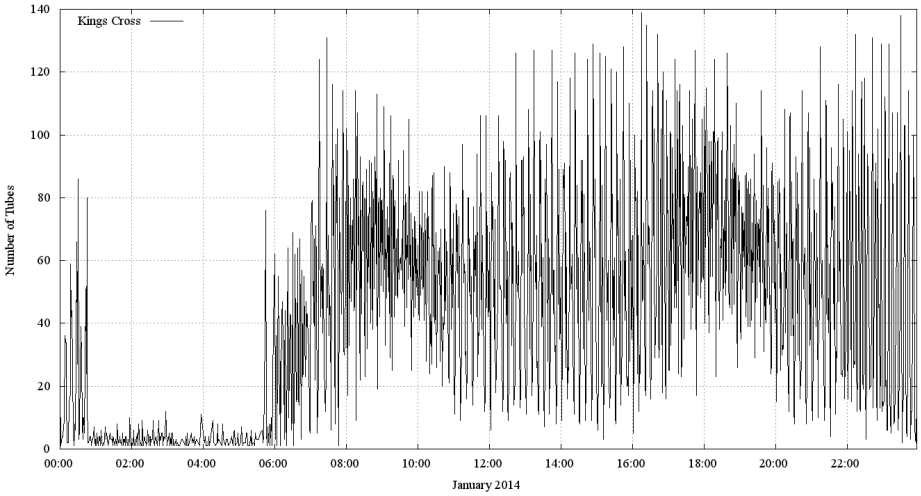


Figure 6.16: Times and number of trains detected as being at Kings Cross as an aggregate of all weekdays in January 2014.

Moving on to the problem of the routes taken for each destination code, this is a simpler problem to solve due to the limited number of routes and the quantity of data. The geographical routes taken by tubes can be obtained by grouping stations by the $DestinationCode$ seen at that location. While this does not order the routes, either the network graph of the tube system can be used (G from algorithm 6), or the timing of arrivals at stations from the real-time API data can also be used to build the station sets into route sequences.

The set of stations forming a route can be determined for any $DestinationCode$ as follows:

$$R(x) = DISTINCT(I(\{StationCode\}), \texttt{where } \ I(\{DestinationCode\}) = x) \tag{6.5}$$

Looking at $R(88)$, which is the Victoria Line code for services to Brixton, the set of stations returned is: $\{BRX, VUX, WAL, KXX, FPK, OXC, TTH, PIM, STK, GPK, SVS, HBY, EUS, BHR, VIC, WST\}$. This is the full complement of stations from one end of the line to the other, which is plotted in figure 6.17. However, destination code 458 returns $R(458) = \{BRX, WAL\}$ as it is a code that has only been used six times during the month to move trains from one end of the line to the other around midnight. This type of anomaly when tubes are being re-positioned makes the analysis of the data challenging.

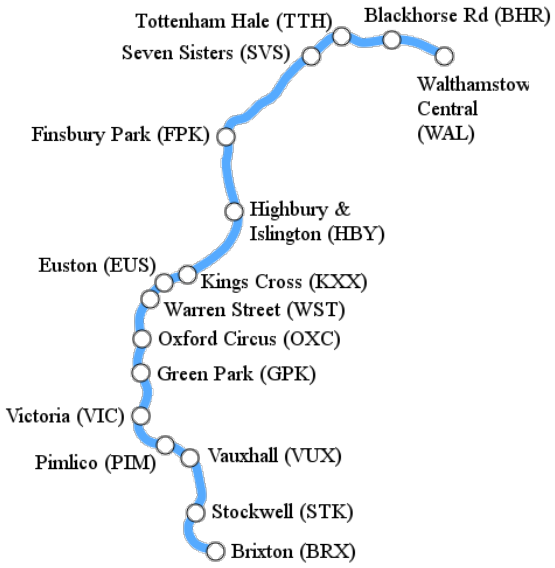


Figure 6.17: The Victoria Line track layout and station codes.

Finally, the distribution of routes used throughout the day at every station on the network is required if the model is to simulate the correct frequency of service. Figure 6.18 shows the distribution of route code usage, while figure 6.19 shows the related distribution of numbers of services passing through every station. This information shows that the actual number of routes in active use on the network is significantly lower than the total number seen in the data.

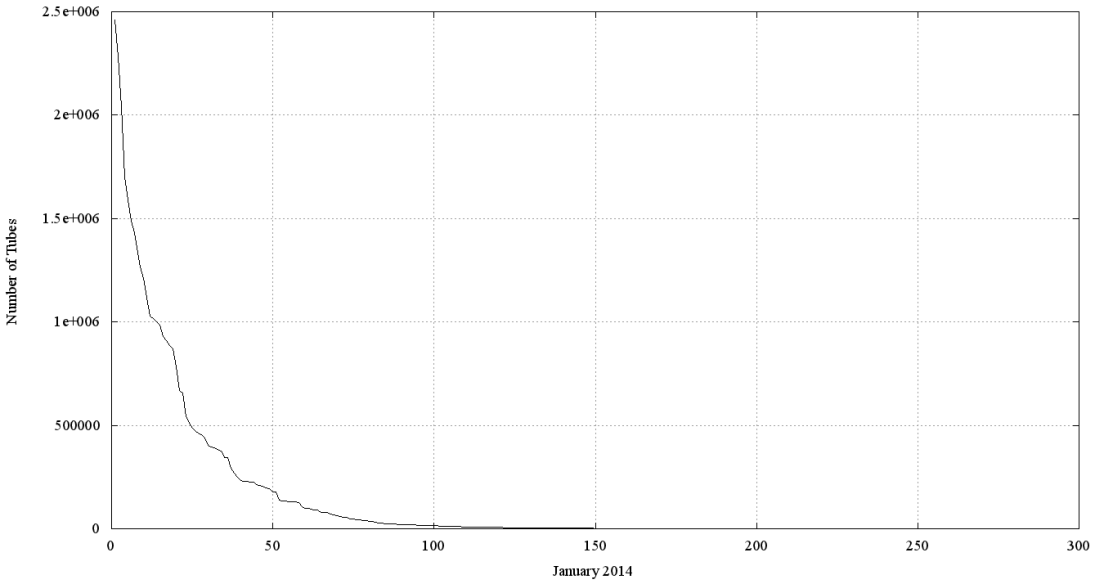


Figure 6.18: January 2014, total number of times each destination code was used. The data is ordered by count (y axis).

Now that the software has been used to examine the real-time data in detail, it is possible to build simulations of the system which can be proved to be statistically similar to the real world to within a controlled level of error. In the model, the decision points in the algorithm can be seen as reducing to a pattern match against previous data. Whether the machine learning part uses an algorithm like '1R', 'Nearest Neighbour' or 'Naive Bayes' is up to the particular implementation. The important point to note is that this technique is designed to build a model to test against and that this is a method for coping with a large quantity of data.

Finally in this chapter, the following two implementations of the technique approach the problem from different directions. Real-time data, by its very nature, requires a communications stream, which suggests an Internet based system. In this context, the model runs on a web page with real-time updates delivered automatically. Being a 'zero-install' solution, but limited by the fact that it has to run inside a web browser, this is suited more to casual users. The main benefit of a web-based solution is to be able

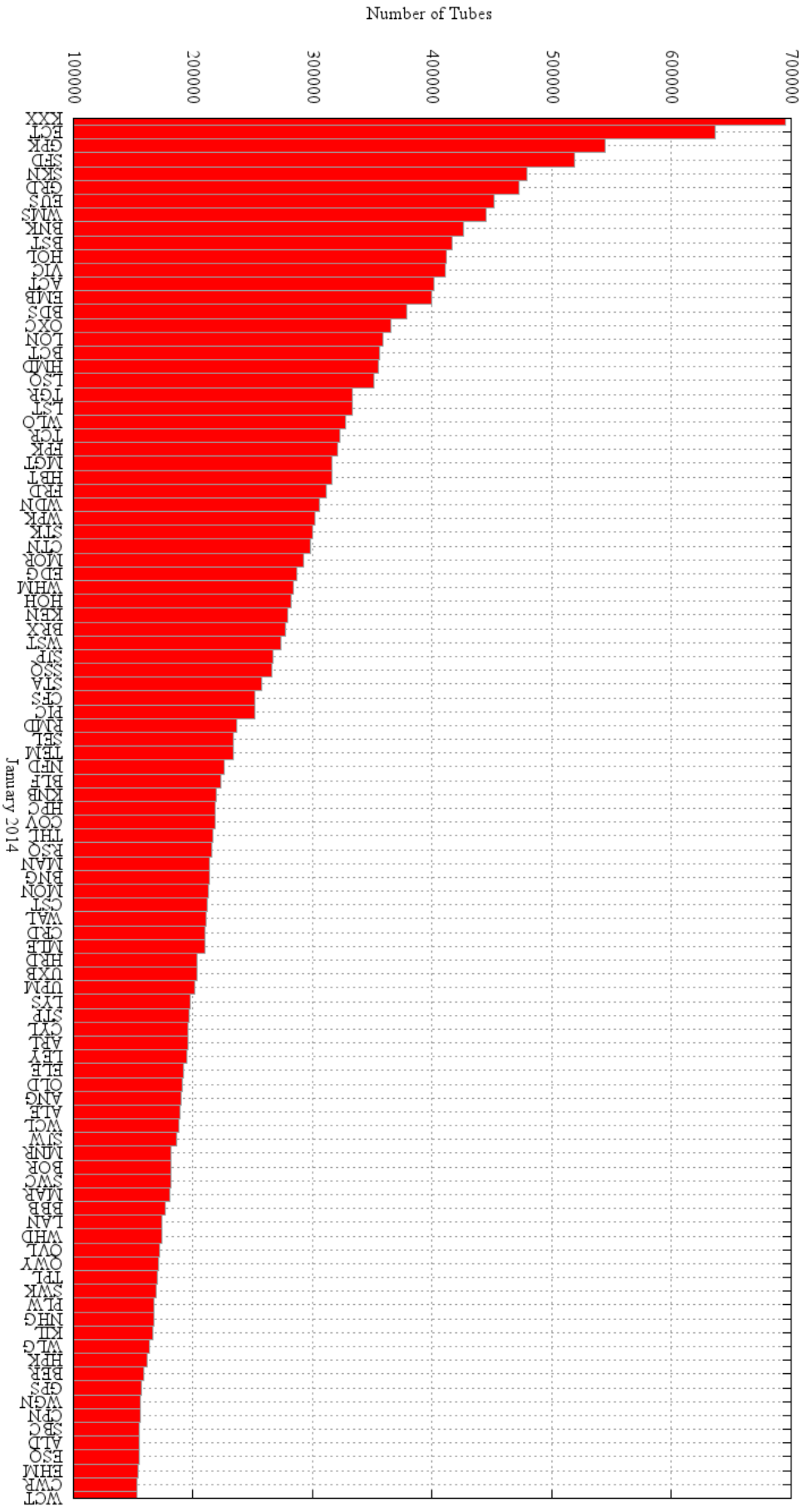


Figure 6.19: January 2014, total number of services passing through stations. The data is ordered by count (y axis).

to share the model with other programmers, much as TfL has shared the London Underground data with developers to build a community of 3rd party applications which benefits everybody. Currently, feature detection with the London Underground data is limited to numbers of tubes running, line and branch disruptions[6], but, by publishing a programmable model of the network, the aim is to provide tools for experimentation and discovery in live city systems. This has been implemented as a new type of MapTube map, using the 'AgentScript' ABM library with a simplified model of the London Underground. Figure 6.20 shows this running and also demonstrates how some obvious, but often overlooked, facts can be re-discovered. In this example, the route choice is a uniform random function and the creation and destruction of tubes has been disabled. The number of tubes on each line does not change, they simply go back and forth forever, picking a random route. This leads to an interesting observation: the spacing between tubes on the Victoria Line does not change, while on all the other lines, the tubes appear to bunch together. Figures 6.21a to 6.21d show this effect in action.

Spacing between trains on any network is technically termed, "headway", which is a feature in the data that would be useful to measure and graph in real-time. The example presented here aims to provide researchers with the tools to perform this kind of 'city diagnostics' research. The answer to the headway problem lies in the structure of the tube network itself. The Victoria Line is the only line with no branches. By starting with the assumption that the tubes are optimally spaced when the system is first initialised with live data, two tubes taking separate route branches will get to the ends of the route, turn around and then converge again when the branches come back together. The headway will now be altered by the difference in time between the two branches, so there is a tendency towards divergence from the optimal running. Although wait times at stations are not included in this model, both this and the network topology serve to upset the spacing and frequency of service. Wait times can be variable and depend on commuter loading of the network during rush-hour, so are typically a high load phenomena of interest to transport systems researchers. How networks perform under high load situations are active areas of research. In the real world, corrections need to be made to the system to keep services running optimally. An interesting question is whether this can be detected in the data and measured in the live system.

While both bus and Network Rail data has been added to MapTube, the limits of

[6] See: *http://citydashboard.org* and other CASA projects like the iPad wall.

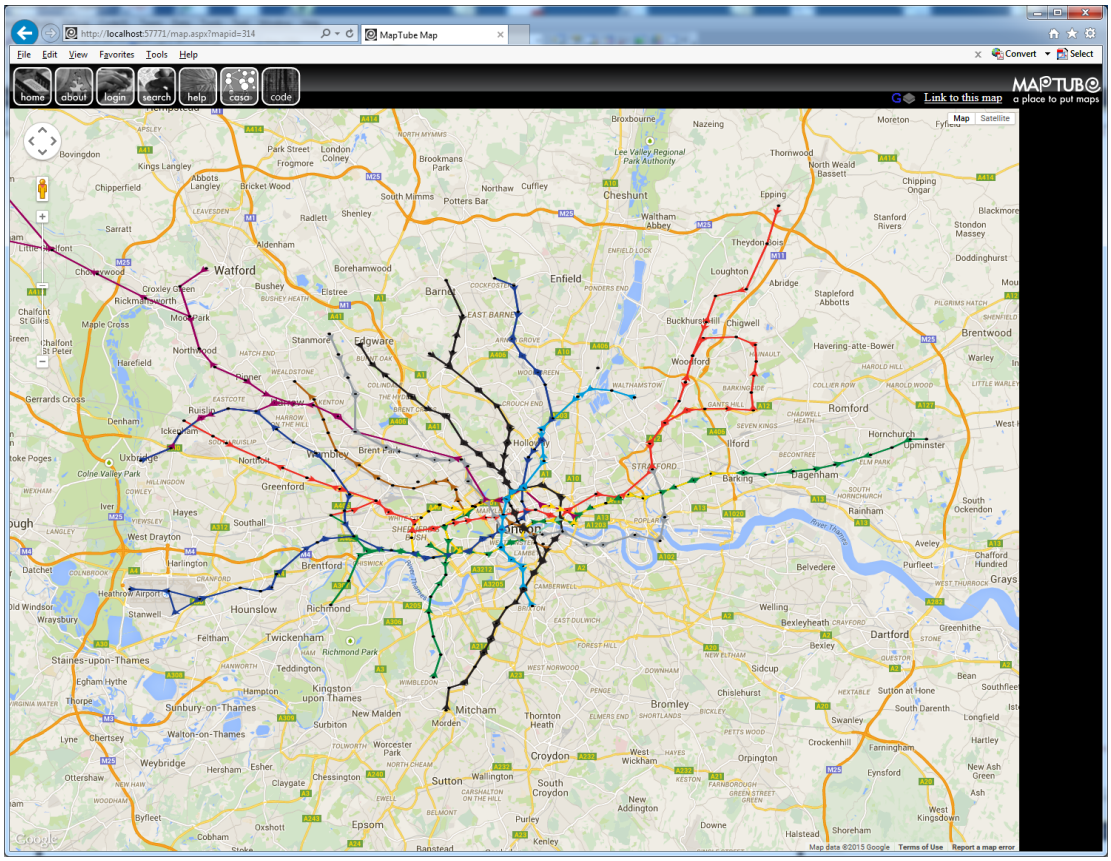
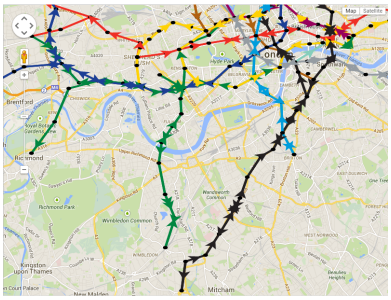
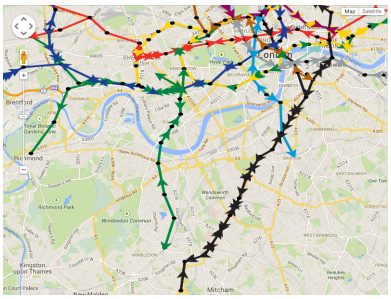


Figure 6.20: The MapTube website showing live animated positions of London Underground trains for 09:50 on Monday 23rd February 2015.

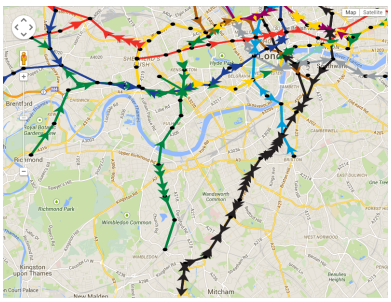


(a) Starting positions.



(b) After 30 minutes.

(c) After 60 minutes.



(d) After 90 minutes.

Figure 6.21: The model is initialised from live data for 20:30 on Monday 23rd February 2015 and then left to run at 30 times normal speed. Times stated are real world times, so simulation run lengths are 60s, 120s and 180s respectively.

the browser become apparent when including data of this quantity. As stated earlier, there are 7,500 buses at peak times, compared to just 450 tubes, and bus routes are more complex with shorter, more frequent stops. The TfL stream for obtaining bus stop locations lists 21,987 separate bus stops, compared to 275 tube stations. For this reason, the second implementation of real-time modelling is the 'GeoGL' virtual globe application, written in C++ and using OpenGL (figure 6.22).

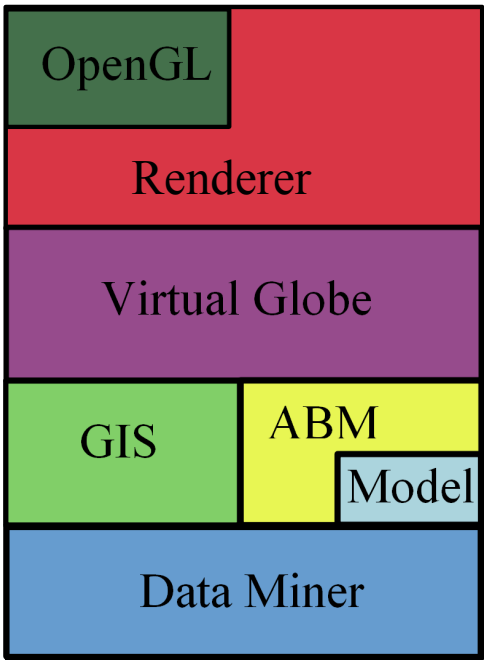


Figure 6.22: GeoGL System Diagram.

6.3.1 3D Rendering, GeoGL Project

This is a development to create a 3D GIS application which is capable of performing complex analysis and visualisation on the quantity and complexity of city data currently being captured. Rather than being a collaborative tool, this is seen more as a data workbench, running the complex forms of the multi-modal transport models and generating exploratory statistics from the simulation which are intended to be analysed using data mining tools.

While figure 6.22 shows how the 'GeoGL' application is structured, figure 6.23 shows the package diagram for the complete project. The structure broadly follows the outline of Cozzi and Ring [CR11] and chapter 3, recognising the 'Need for a Renderer', to abstract the low level device operations away from the virtual globe, GIS and ABM functions.

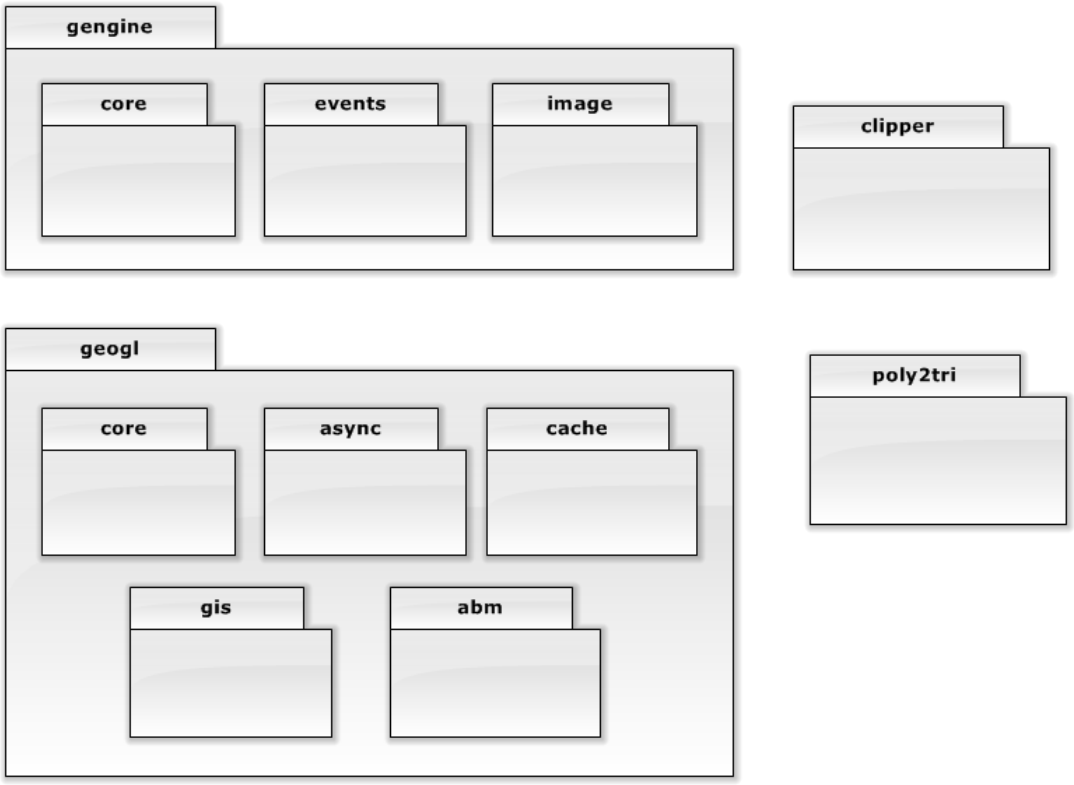


Figure 6.23: The packages comprising the GeoGL project.

The intention here is to build a system which is graphics device agnostic, so, in theory, the OpenGL layer could be switched for DirectX[7] and fall back strategies for systems without programmable shaders can be implemented with all the changes confined to the renderer layer. Rule number one for coding the virtual globe layer is, "No OpenGL Calls", so total isolation with everything handled via the renderer layer. The key requirement here is to allow enough flexibility at the virtual globe level to be able to investigate some of the more interesting rendering issues which are open research questions in this type of large scale temporal data visualisation. In their book on virtual globe systems, Cozi and Ring briefly mention the idea of rendering the globe with the shader rather than polygons [CR11, pp149] and touch on the topic of tessellation, commenting on Miller and Gaskins' globe tessellation algorithm [MG09]. The problem here concerns the polar regions, where the tessellation is extreme, causing artefacts in the rendering. Alternatively, the agent rendering itself is another area of interest, due to their potentially high number and the fact that they are moving on a piecewise linear surface, rather than a true curved surface. One approach to rendering high numbers of agents is to pre-render 2D imposters, as in [TLC02], an idea which has

[7]DirectX is ©Microsoft.

some merit when dealing with geometric shaped agents (cube, sphere, arrow etc.) at distance. Sheffield University's "FlameGPU" library is designed for large-scale, high performance rendering of agent based models [She19]. In the publication, "Resolving Conflicts between Multiple Competing Agents in Parallel Simulations" [Ric14], Richmond tackles the problem of conflict resolution in a parallel model. With any optimisation by parallelisation where the instructions can complete out of order, the results between model runs can vary, even where the conditions are identical, so this needs to be addressed.

The graphics engine structure is shown in the class diagram in figure 6.24. The structure of this part of the package is intended to be an abstraction between the graphics API, which is OpenGL here, and the rest of the software, which was a key component of the design.

As was mentioned previously, low level access to the hardware was required, so the implementation is nothing more than a lightweight wrapper around the graphics hardware. In terms of risk management for the project, it was always an idea to be able to swap out the renderer layer for a more mature project like Ogre, which is a more specialist game engine. This is not something that has been investigated, but the ABM layer has also been designed to run without the virtual globe by using the rendering engine directly. This is achievable because of the isolation level that has been maintained between the different sections of the project shown in figure 6.22.

The next layer in the diagram is the 'virtual globe' layer. The class diagram in figure 6.25 shows how this is structured. The requirement here is to be able to render an Earth which is textured to a level of detail equivalent to the NASA Blue Marble images and containing geometry for buildings, rivers and roads, in addition to physical manifestations of agents and links from the ABM layer.

The main class is the 'Globe', which uses a 'TiledEarth' object to render the textured portion of the globe currently visible on the screen. This is the ellipsoid from the WGS84 specification, so world coordinates are geodetic latitudes and longitudes. Due to the enormous change in level of detail from the space view of the Earth down to street level, the 'TiledEarth' uses a system of re-meshing portions of the Earth based on the current viewport. The 'Tau' factor determines when a split happens, which results in a patch being split into four child patches and added to the scene. This is a 'chunked level of detail' system following the work presented in [Ulr02]. An important point to

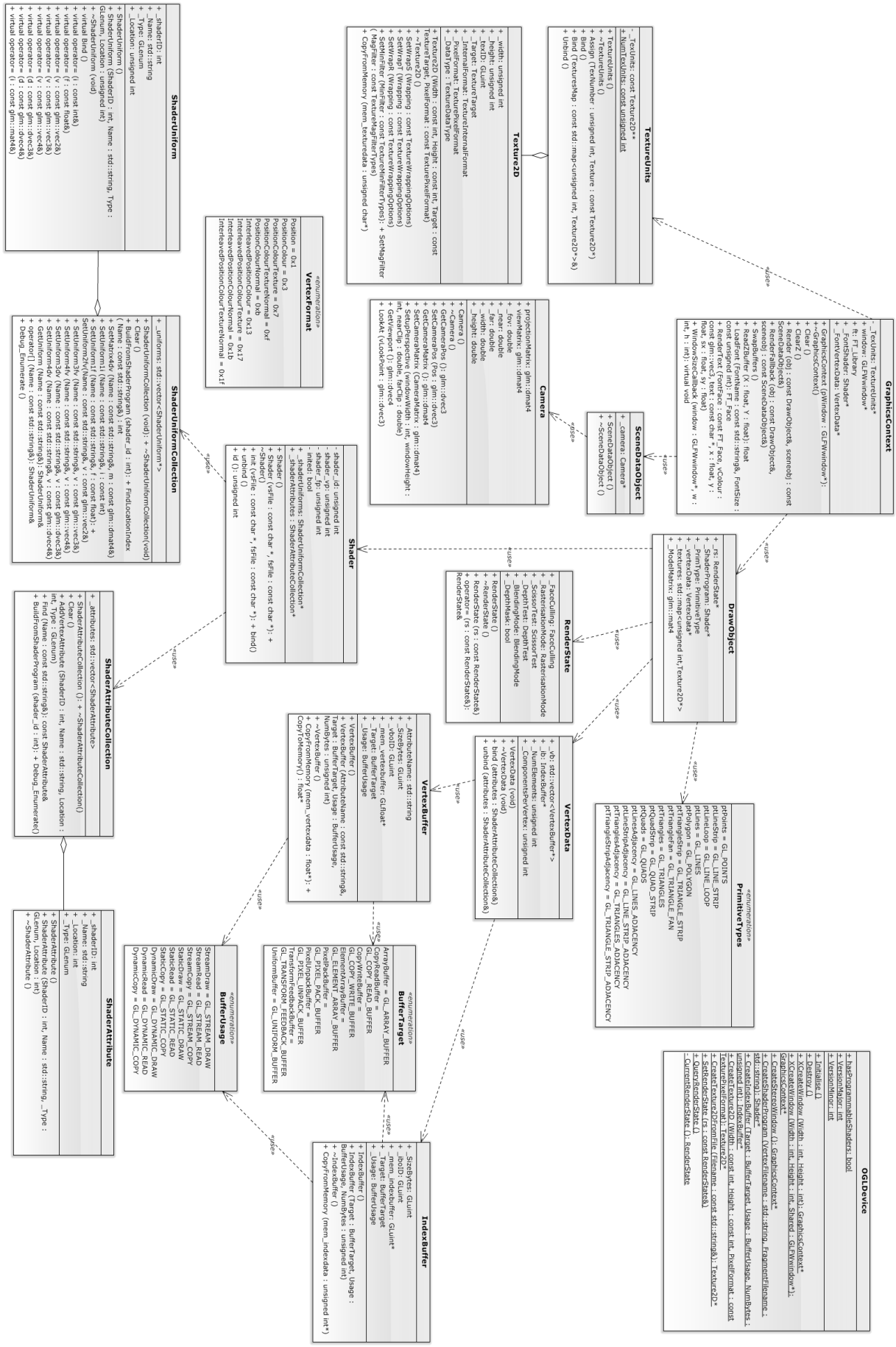

GraphicsContext
- _TexUnits: TextureUnits*
- _MainWindow: GLFWwindow*
+ FT: FT_Library
- _FontShader: Shader*
- _FontVertexData: VertexData*
+ GraphicsContext (pWindow : GLFWwindow*);
+~GraphicsContext()
+ Clear ()
+ Clear2 ()
+ Render (obj : const DrawObject&, sceneobj : const SceneDataObject&)
+ Render (obj : const DrawObject&, scene : const SceneDataObject&)
+ SwapBuffers ()
+ ReadzBuffer (X : float, Y : float): float
+ LoadFont (FontName : const std::string&, FontSize : const unsigned int): FT_Face
+ RenderText (text : const char *, x : float, y : float, sx : float, sy : float)
+ WindowSizeCallback (window : GLFWwindow*, w : int, h : int): virtual void
«use»
TextureUnits
- _TexUnits: const Texture2D**
± NumTexUnits: const unsigned int
+ TextureUnits ()
+ ~TextureUnits ()
+ Assign (TexNumber : unsigned int, Texture : const Texture2D*)
+ Bind ()
+ Bind (TextureMap : const std::map<unsigned int, Texture2D*>&)
+ UnBind ()
«use»
SceneDataObject
+ _camera: Camera*
+ SceneDataObject ()
+ ~SceneDataObject ()
«use»
DrawObject
+ _FS: RenderState*
+ _ShaderProgram: Shader*
+ _PrimType: PrimitiveType
+ _VertexData: VertexData*
+ _IndexBuffer: unsigned int,Texture2D*>
+ _ModelMatrix: glm::mat4
«use»
«use»
«use»
«use»
«enumeration»
PrimitiveTypes
ptPoints = GL_POINTS
ptLineStrip = GL_LINE_STRIP
ptLineLoop = GL_LINE_LOOP
ptLines = GL_LINES
ptPolygon = GL_POLYGON
ptTriangleStrip = GL_TRIANGLE_STRIP
ptTriangleFan = GL_TRIANGLE_FAN
ptTriangles = GL_TRIANGLES
ptQuadStrip = GL_QUAD_STRIP
ptQuads = GL_QUADS
ptLines4Adjacency = GL_LINES_ADJACENCY
ptLineStrip4Adjacency = GL_LINE_STRIP_ADJACENCY
ptTriangles4Adjacency = GL_TRIANGLES_ADJACENCY
ptTriangleStrip4Adjacency = GL_TRIANGLE_STRIP_ADJACENCY
OGLDevice
+ hasProgrammableShaders: bool
+ VersionMajor: int
+ VersionMinor: int
+ Initialise ()
+ Destroy ()
+ XCreateWindow (Width : int, Height : int): GraphicsContext*
+ XCreateWindow (Width : int, Height : int, Shared : GLFWwindow*): GraphicsContext*
+ CreateStereoWindow (): GraphicsContext*
+ CreateShaderProgram (VertexFilename : std::string, FragmentFilename : std::string): Shader*
+ CreateIndexBuffer (Target : BufferTarget, Usage : BufferUsage, NumBytes : unsigned int): IndexBuffer*
+ CreateIndexBuffer (Target : BufferTarget, Usage : BufferUsage, NumBytes : unsigned int, IndexData : const int, Height : const int, PixelFormat : const TexturePixelFormat): Texture2D*
+ CreateTexture2DFromFile (Filename : const std::string&): Texture2D*
+ SetRenderState (rs : const RenderState&)
+ QueryRenderState ()
- _CurrentRenderState (): RenderState
Texture2D
- _width: unsigned int
- _Height: unsigned int
- _Target: TextureTarget
- _InternalFormat: TextureInternalFormat
- _PixelFormat: TexturePixelFormat
- _DataType: TextureDataType
+ Texture2D (Width : const int, Height : const int, Target : const TextureTarget, PixelFormat : const TexturePixelFormat)
+ ~Texture2D ()
+ SetWrapS (Wrapping : const TextureWrappingOptions)
+ SetWrapT (Wrapping : const TextureWrappingOptions)
+ SetMagFilter (MagFilter : const TextureMagFilterTypes)
+ SetMinFilter (MinFilter : const TextureMinFilterTypes)
+ SetMagFilter (MagFilter : const TextureMagFilterTypes)
+ CopyFromMemory (mem_texturedata : unsigned char*)
Camera
+ projectionMatrix: glm::dmat4
+ viewMatrix: glm::dmat4
- _near: double
- _far: double
- _width: double
- _height: double
+ Camera ()
+ ~Camera ()
+ GetCameraPos (): glm::dvec3
+ SetCameraPos (Pos : glm::dvec3)
+ GetCameraMatrix (): glm::dmat4
+ SetCameraMatrix (CameraMatrix : glm::dmat4)
+ SetupPerspective (windowWidth : int, windowHeight : int, nearClip : double, farClip : double)
+ LookAt (viewFront : glm::dvec3)
RenderState
+ _FaceCulling: FaceCulling
+ _RasterisationMode: RasterisationMode
+ _StencilTest: StencilTest
+ _BlendingMode: BlendingMode
+ _DepthMask: bool
+ RenderState ()
+ ~RenderState ()
+ RenderState (rs : const RenderState&)
+ operator= (rs : const RenderState&): RenderState&
VertexData
+ _vb: std::vector<VertexBuffer*>
+ _ib: IndexBuffer*
+ _NumElements: unsigned int
+ _ComponentsPerVertex: unsigned int
+ VertexData (void)
+ ~VertexData (void)
+ bind (attributes : ShaderAttributeCollection&)
+ unbind (attributes : ShaderAttributeCollection&)
«enumeration»
VertexFormat
Position = 0x1
PositionColour = 0x3
PositionColourTexture = 0x7
PositionColourTextureNormal = 0xf
PositionColourNormal = 0xb
InterleavedPositionColour = 0x13
InterleavedPositionColourTexture = 0x17
InterleavedPositionColourNormal = 0x1b
InterleavedPositionColourTextureNormal = 0x1f
ShaderUniformCollection
+ _uniforms: std::vector<ShaderUniform*>
+ ShaderUniformCollection (void): + ~ShaderUniformCollection(void)
+ Clear ()
+ BuildFromShaderProgram (shader_id : int): + FindLocationIndex (Name : const std::string&) : int
+ SetMatrix4fv (Name : const std::string&, m : const glm::dmat4&)
+ SetUniform1i (Name : const std::string&, i : const int);
+ SetUniform1f (Name : const std::string&, f : const float); +
SetUniform2fv(Name : const std::string&, v : const glm::vec2&)
+ SetUniform3fv (Name : const std::string&, v : const glm::vec3&)
+ SetUniform4fv (Name : const std::string&, v : const glm::vec4&)
+ SetUniform1d (Name : const std::string&, v : const glm::dvec2&)
+ SetUniform3dv (Name : const std::string&, v : const glm::dvec3&)
+ SetUniform4dv (Name : const std::string&, v : const glm::dvec4&)
+ operator[] (Name : const std::string&): ShaderUniform&
+ Debug_Enumerate ()
«use»
Shader
- shader_id: unsigned int
- shader_vp: unsigned int
- shader_fp: unsigned int
- inited: bool
+ _shaderUniforms: ShaderUniformCollection*
+ _shaderAttributes : ShaderAttributeCollection*
+ Shader ()
+ Shader (vsFile : const char *, fsFile : const char *): +
~Shader()
+ init (vsFile : const char *, fsFile : const char *): + bind()
+ unbind ()
+ id (): unsigned int
«use»
VertexBuffer
+ _AttributeName: std::string
+ _SizeBytes: GLuint
+ _mem_vertexbuffer: GLfloat*
+ _Target: BufferTarget
+ _Usage: BufferUsage
+ VertexBuffer ()
+ VertexBuffer (AttributeName : const std::string&, Target : BufferTarget, Usage : BufferUsage, NumBytes : unsigned int)
+ ~VertexBuffer ()
+ CopyToMemory (mem_vertexdata : float*): + CopyToMemory(): float*
«use»
«use»
«use»
«use»

Figure 6.25: GeoGL class diagram.

note is that all the patches comprising the Earth mesh share the same index buffer. Only the vertices and texture offsets are changed as a result of re-meshing. While alternative techniques for tessellating the Earth exist, for example, cube map deformation [MG09] and GPU ray casting [CR11, Ch4.3], the approach taken here is to use the standard geographic grid tessellation. While this results in some visible artefacts near the poles where all the grid lines meet at a point, most of the data being visualised will be city based and mid-latitude, so this is unlikely to be an issue[8].

Fortunately, GPUs with 64 bit floating point instructions are now common, so numerical precision was not a critical issue. Based on the WGS84 definition, the Earth spheroid has a semi-major axis of 6378137 metres and semi-minor axis of 6356752 metres. Taking 6378137 metres as a single precision floating point number, the next representable point is at 6378137.5 metres, giving a 0.5 metre precision at this point on the Earth's surface at the equator. With 64 bit precision, the next point is 6378137.000000000931, which eliminates the 'jitter' problems which occur with the single precision rendering and rounding of points to the nearest representable value. There is a unit test called 'fp-precision_test' as part of the project which establishes these limits.

However, while points can be represented with 64 bit precision, the GPU's depth buffer only has 24 bits. The standard approach to this problem is to use multi-frustum rendering. The technique is to partition the scene by distance from the viewer, using 1,000,000,000 metres to 1,000,000 metres for the first pass, then 1,000,000 metres to 1,000 metres for the second pass, then 1,000 metres to 1 metre for the final pass. The Z buffer is cleared between each pass, with 'Painter's Algorithm' used to aggregate each frustum pass into a single image. Problems with this technique occur when objects lie between these cut-off points and when there is transparency on the object. In practice, the system is using these limits, but with a small amount of overlap between them. A similar approach is taken in the Cesium[9] library for WebGL and NASA's World Wind.

Caching is one of the most important topics in any virtual globe, due to the huge quantity of information available. A large part of this project implements a caching and work flow system designed to provide the rendering part with the information needed to draw the current view. Starting with the lowest latency medium, data exists in memory, on a local disk cache, on a remote disk cache and finally, at the highest latency, on the

[8] Replacing the Earth tessellator is simply a matter of writing a replacement class and would make for an interesting project in its own right.

[9] Cesium website for Javascript globes: *http://cesiumjs.org*.

Internet. This means that a request for a building or for a new patch of the Earth that has just come into view for the first time, is unlikely to be satisfied within a single frame update. The class diagram in figure 6.26 shows the local disk cache structure.

Requests for resources are made through this interface, which maintains a 'least recently used' (LRU) store of files. When a request for a new resource is made, if that resource is currently available in the cache, then the return result will indicate this and the resource can be instantiated for rendering immediately. If the resource is not yet available, then a negative response is returned and the rendering thread continues without it (non-blocking requests). In the background, though, the cache will begin trying to acquire the resource locally, so that, some time in the future, a request for the resource will return true. The reason for using this type of non-blocking request is that, with the view moving around rapidly, the resource might no longer be in view when it is eventually available several hundred, or even several thousand frames later. How a virtual globe implements this type of background work flow system can have a large bearing on the overall performance. Also, predicting what is about to come into view and pre-emptively loading resources is a valid technique and an open area of research, although probably too complex to consider implementing here.

Acquiring resources is only the first part of the work flow, though. Figure 6.27 shows the most complicated part of the GeoGL project, which is the asynchronous handling of resource loading and background mesh and buffer creation. This is implemented as a simple message queue pattern, similar to the method outlined in [CR11, Ch10], rather than using a full 'Mediator/Observer' or 'Publish/Subscribe' pattern [Gam+99] which have been used in other Javascript based 3D systems[10]. The message passing system used does have the added feature of being able to run the code single-threaded which greatly simplifies the debugging process. The 'DataCache' creates a message queue, onto which it attaches a worker object. When it wants to load a resource it posts a 'load' message to the queue containing the address of the resource to load and where to store it in the local cache. The worker does the work on a background thread, posting a completion message to a 'done' queue when it is finished. Newly acquired resources are picked up off of this queue and are then pushed on to a 'MeshWorker' queue to be turned into a 3D mesh and vertex buffers for rendering. This method of using split queues for file I/O and GPU operations keeps all the very long running

[10]The beta version of the ViziCities project originally used a publish subscribe pattern which was later changed.

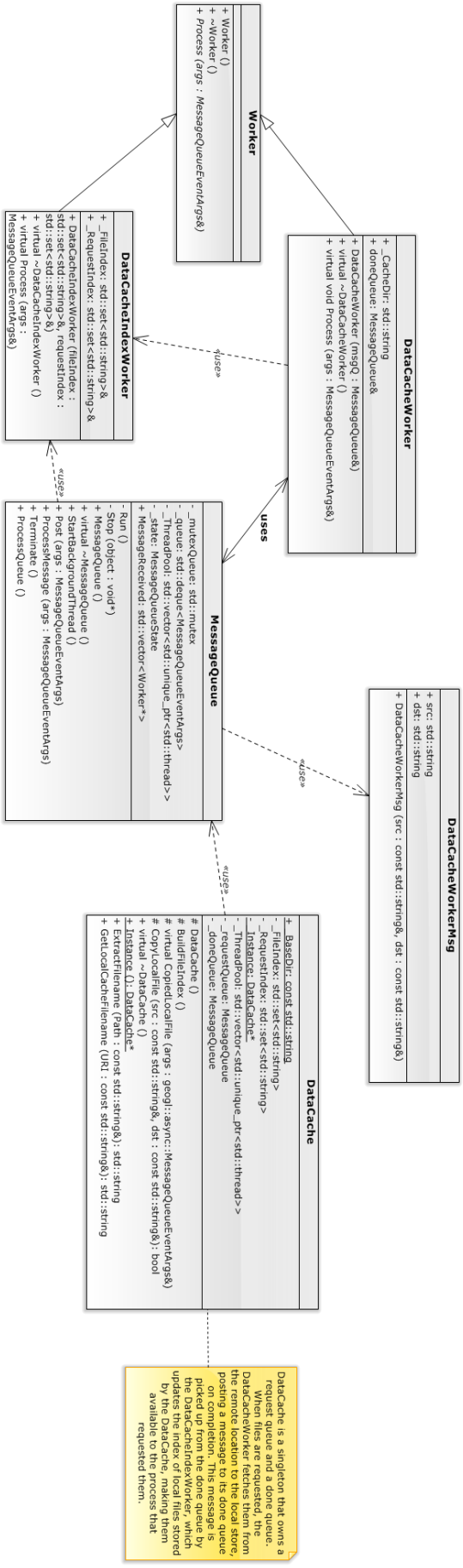


Figure 6.26: GeoGL Cache class diagram.

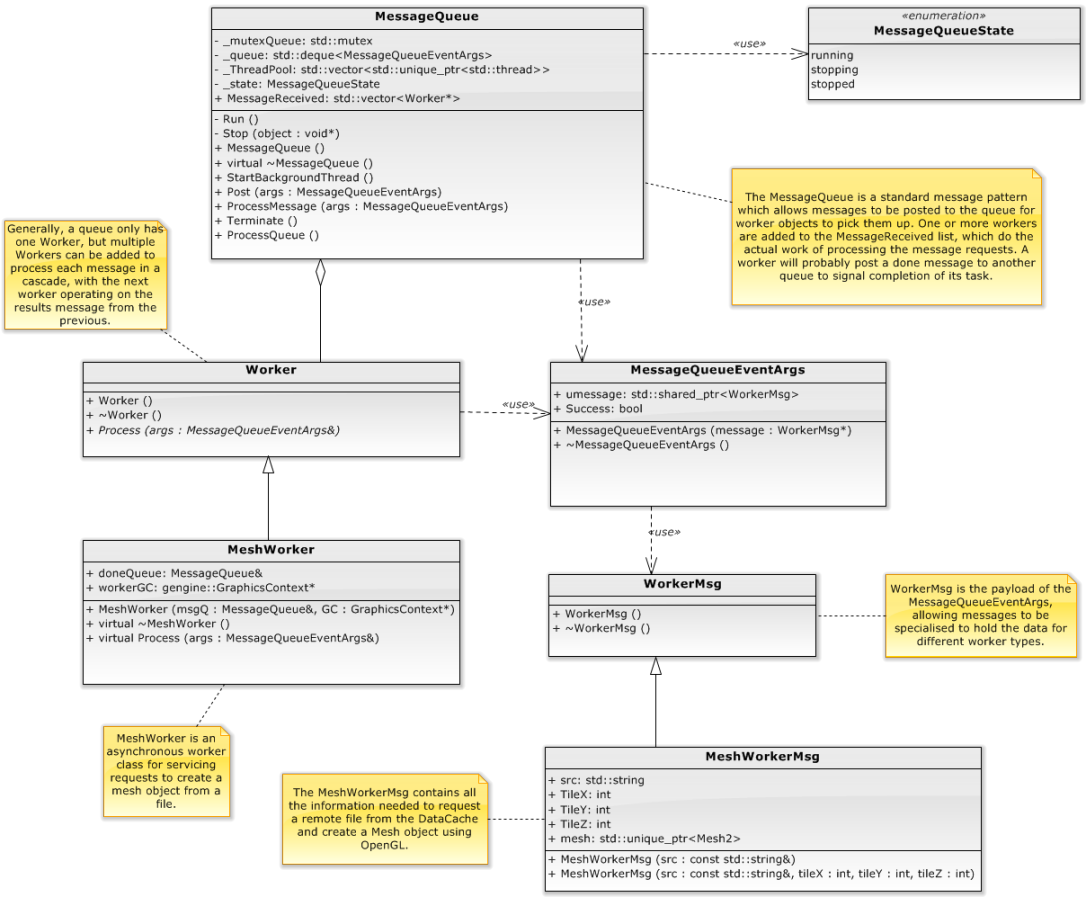
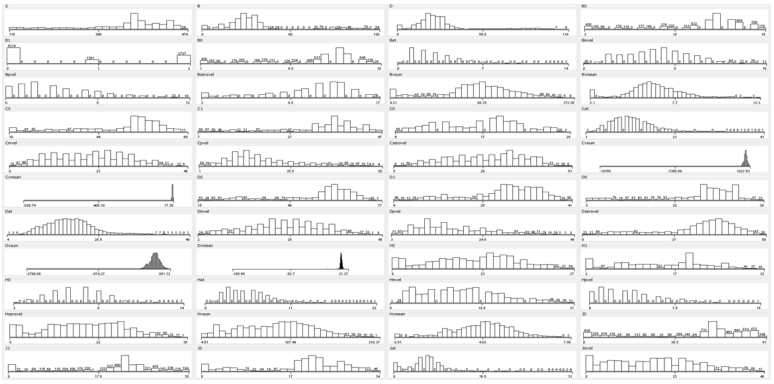


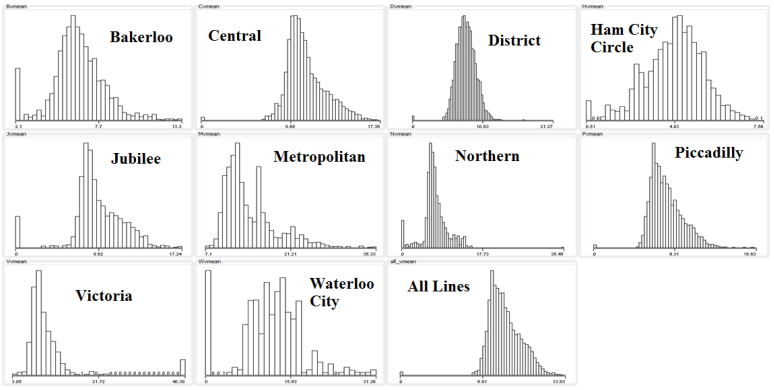
Figure 6.27: GeoGL Async class diagram.

Operating System related file I/O contained away from the other tasks which have a much faster turnaround time.

The final element to the system is the ABM layer, which is where instrumentation becomes possible. The analysis of the London Underground model is the key focus of these initial stages of mining the data source for information. In the charts shown in figure 6.28b, graphs of the speeds of tube trains on each of the lines has been extracted from the data and plotted as a probability density. Experimentation using the model requires the addition of a function to calculate and log an interesting feature for analysis offline. In the London Underground example, the aim is to find features in the data which can be used as indicators of problems in the service. The graphs in figure 6.28a show a combination of factors including: number of running services, number of tubes at station, number of tubes between stations, average time to station, average speed, number of trains running faster or slower than the segment runlink time, number created, number destroyed and the maximum, minimum and average headway. This instrumentation is the subject of the next chapter on data exploration.



(a) Graphs of data features being monitored by the tube model. These are a combination of speed, direction up/down ratio, zero velocity trains and over or under expected speed.



(b) Average line speed data extracted from the model output.

Figure 6.28: The two figures above show the data extracted from running the tube model using 24 hours of data from the tube strike beginning on 28th April 2015 at 9pm.

6.3.2 Conclusion

Returning to figure 6.4, "Agent Based Modelling (ABM) meets Machine Learning", earlier in this chapter, the model uses information from the real-time data stream to ground the simulation, maintaining the link between the real world and the simulated world, while allowing measurements to be made on the system while it is running. Timetable data has deliberately not been used, as the primary goal of this work is to be able to analyse the system when everything has gone wrong and the timetables are irrelevant, for example during the strike days. The comparison between the model and the current real-time data is used to highlight any abnormalities. If, for example, the number of District Line trains on a segment of the network started to drop due to a signal failure causing service disruption, then this can be detected in real-time and highlighted, which is an applied example of knowledge directed visualisation. Conversely, the simulation can be used to experiment on, so, for example, failures can be deliberately introduced into the system, allowing researchers to experiment on a real-time city using live data. While this might be of limited use for the London Underground net-

work in isolation, the power of this technique is to apply it to bus data, Network Rail, Highways Agency and Oyster card data to build a connected multi-modal network that allows for complex analysis using models of both vehicles and commuter behaviour.

The London Underground system was chosen in this chapter as a starting point, being the simpler of the bus, train and private transport networks. The example was chosen because it fulfils all the criteria of real-time, noisy data and complex information analysis, resulting in a complex dynamic geographical system. The following chapter expands on this work and includes models of the remaining transport networks built directly from the archives of real-time data currently accumulated. This is the beginnings of a real live inhabited city.

6.4 Static and Real-time

While the real-time data presented so far has relied on data from APIs, the background data requires the same type of static map as the 2D web based mapping presented in the previous chapter. This is where the two systems come together, with the 2D mapping providing cartography for the 3D globe visualisation which gives a sense of place. Due to the huge quantity of data, tiling and hierarchical level of detail applies to both the 2D and 3D mapping, the only problem being the projection system being used. With the advent of vector tiling, systems exist which can supply data to a 3D system at an appropriate level of detail. For a 2D mapping system, this would generally use a Spherical Mercator projection and tile grid. The 3D systems would naturally prefer data to be in a geographic coordinate system like WGS84, as it makes the conversion to the Cartesian 3D system simpler. This is not a huge problem, though, as a WGS84 tiling system could be created, or the Mercator data reprojected on the fly.

In addition to cartography like roads, rivers and railways, data layers can also be included. Returning to the concept of the ‘MapTubeV’ vector tiler explored in the previous chapter, the need to add shapefile loading functions to the 3D visualisation is now redundant. The 3D visualisation software can now concentrate on just that and leave most of the other GIS functions to an array of web services. This is already required by the real-time visualisation, so, by extending it to the static data as well, the system becomes more like a distributed GIS than a traditional desktop GIS. It also gains the ability to handle a higher quantity of data in a more scalable way, using components that can be swapped in or out and extended as needed in a truly heterogeneous way.

The key to this system is the interoperability between components, which in turn depends on geospatial standards. With the OGC's WMTS standard available to provide tiles in different projections for both 2D and 3D systems, along with WFS to provide vector data in GML within a given bounding envelope, data providers can be built from a variety of components. The GeoGL current system uses the OS 3D building footprint data from their Open Data release, built using a custom written Java based tiling system. This provides vector tiles in the 3D graphics, Wavefront OBJ format, which is ubiquitous in computer graphics. The design is intended to be as architecture neutral as possible, enabling different systems to be plugged together to solve problems, but in doing so to treat all data as usable in the simulation. As an example, the OS building footprint data is used in a number of 3D visualisation systems to give the appearance of a 3D city, but without being anything other than 'decoration'. It is fairly easy to imagine building heights, weather, air quality and transport data all coming together in a visualisation of NO_2 levels based on wind direction and street canyon layout. Data stores can also be seen as sources of information. With support for vector formats added to the MapTube site, data on population density or travel to work data can be combined with real-time transport information as it happens.

6.5 Conclusion

This chapter and the previous one are titled 'Dynamic Visualisation' and 'Real-time Simulation and Agent Based Models' because they build up a framework of tools which are designed for exploring relationships between geospatial datasets, handling real-time information and data exploration. Static data and real-time streams have been explored, using a heterogeneous approach designed as a framework of 'plug and play' services which can be combined together to make complex analysis work flows. Both 2D and 3D visualisation has been explored, highlighting the need for greater interoperability between geospatial data formats. How data is passed between systems is still a problem, with a split between data visualisation which works on a geographically local subset as distinct from cartography which handles large scale data for the whole world. While the OGC's 'Web Feature Service' (WFS) can return geographic data within a bounding box, the migration of 3D games technology into browsers has meant that web based mapping has moved from image tiling to vector tiling. The 'MapTubeV' vector tiler presented in the previous chapter is an attempt to bridge this gap, being able

to supply geographic data to both 2D and 3D systems at an appropriate level of detail. This is more like the clumped level of detail hierarchies used in 3D rendering systems like Google Earth and is not something that has previously been part of any geospatial standard. The merging of 2D visualisation with 3D is something that is likely to increase, so solutions to the interoperability problem need to be found.

When real-time city data is included, then data exploration is as much about work flows as about the analysis itself. With APIs giving access to what is happening now and large data stores of static information available to researchers, new tools which can handle these types of data need to be developed. The emphasis in these two chapters has been to create projects which plug the gaps between systems and so provide a fully connected approach to city scale data acquisition, analysis, visualisation and simulation.

The next chapter looks at the problems which the systems developed in this chapter can be used to solve.

Chapter 7

Data Exploration: Data Stores and Correlation

While the previous chapter introduced examples covering connected data, real-time data and data exploration, this chapter expands on those ideas and applies the techniques in the context of current research to determine whether the methodology leads to new discoveries. The application of the techniques to the data analysis task is covered here, with performance analysis, metrics and any optimisations made to the algorithms in order to make them possible on the available hardware included where necessary.

To quote Donald Knuth, speaking about the balance between pure and applied computer science in 1977,

> "It seems to me that both theory and practice are essential, and my life's work has been to keep blending them with each other. You can't get very far in practical work without abstract theories that permit you to think at a higher level, and at the same time theoretical work becomes dead if it doesn't receive fresh inspirations from practical problems in the'real world.' "
>
> (D. Knuth, 1977, [Knu96, pp125])

This is the applied part.

7.1 Data Stores and Correlation

Returning to the linked data example outlined in section 5.2, where data from the 2011 Census was used to correlate variables geographically, performance issues prevented the complete analysis of every variable combination. There are two solutions to this problem: use faster hardware or use a more efficient algorithm (or a combination of the two). Despite this being an inherently data parallel problem, the data and performance metrics gained from the simple example suggests using a more efficient algorithm. While *MapReduce* architectures like Hadoop and commercial cloud computing would

significantly speed up performance on what is a parallel problem, the step from MSOA level data ($A = 7201$) to LSOA ($A = 34753$) and OA ($A = 181408$) data requires $A^2 = 7201^2 = 51,854,401$ (MSOA), $1,207,771,009$ (LSOA) and $32,908,862,464$ (OA) operations inside the correlation loop respectively.[1] Taking the data from the graph in figure 5.18, which shows the correlation value (I) for the population table (KS101) against every other Census variable, the majority of data around zero can be discarded, leaving only the positive and negative tails. This suggests that the naive approach to data mining in this instance is wasteful of computer resources, with only a small percentage of the data leading to useful results. Using the *'87 days for all datasets using one processing element'* calculation from chapter 5.2 as an approximate guide, 100 cloud computing cores running the same algorithm could be reasonably expected to complete within a day[2], as the Amazon reference hardware is similar in performance to the reference hardware used here. Scaling this up to the LSOA level is an increase in computing by a factor of $\frac{34753}{7201} \approx 4.8$, while OA level is a factor of $\frac{181408}{7201} \approx 25.2$, leading to 2,500 cloud compute nodes being required to perform the calculation in one day. Although one possible optimisation might be to compute all the correlations at MSOA level and then sub-divide into OA areas only for highly positive or negative results, many alternative heuristics are also possible. The rest of this section explores some of these heuristics with a view to finding a geospatial computing technique which allows comparisons to be made to establish linking between datasets as a general geospatial data mining technique that can be performed on the desktop. Key features are the ability to handle data from different geographies and linear scalability when adding a new dataset to an existing set of linked data.

7.1.1 Census 2011 Analysis

Before performing any analysis, it is necessary to first establish some general properties related to map comparisons. For two choropleth maps, M_1 and M_2, based on the same geography, each is a vector of data values where $M_1 = \{X_0, X_1, X_2 \dots X_n\}$ and $M_2 = \{Y_0, Y_1, Y_2 \dots Y_n\}$. Elements X_i and Y_i always represent the same geographic area on the map. Maps M_1 and M_2 can potentially have missing values, but, as this does not happen in the 2011 Census data, the handling of missing values is ignored for the

[1] These figures are based on the formula from equation 5.4, although, in practice, the correlation is symmetric, so the actual number of operations could be halved.

[2] This calculation is based on the fact that the Amazon Cluster Compute reference hardware is a dual socket Intel Xeon 5570 with four cores per socket running at 2.93GHz [HP11, pp458, figure 6.15]. The reference hardware used here is a dual core 2.4GHz Pentium Core 2 Duo P8600 with 4GB RAM.

moment. Where choropleth maps are concerned, classification of the raw X and Y values into a finite number of classes will have been performed, possibly using a Fisher Jenks classification [Fis58] [Jen77], uniform breaks or quantiles. Another term for this process is 'stratification'. In spatial correlation, each area has a position, usually the centroid of the geographic area, which gives rise to a physical distance between every pair of centroids. If the centroids are given by $C = \{\underline{c}_0, \underline{c}_1, \underline{c}_2 \dots \underline{c}_n\}$, where $\underline{c}_i$ is a 2D vector of East and North coordinates, then the weights matrix can be defined as $W_{ij} = f(\underline{c}_i, \underline{c}_j)$. The actual function $f()$ which converts distance into weight could be defined simply as $1/distance$, but some alternative functions are covered later in the chapter.

Now that the datasets have been defined, the similarity between two maps can be expressed as:

$$I_{ij} = M_i \otimes M_j \tag{7.1}$$

The following are some of the important properties of the correlation operator, $\otimes$, applied to map similarity:

- Autocorrelation on the leading diagonal, $I_{ii} = I_{jj} = I_{kk} = 1$
- Commutative, $I_{ij} \equiv I_{ji}$, or $M_i \otimes M_j \equiv M_j \otimes M_i$
- Not transitive, so if $I_{ab} = I_{bc}$ then I_{ac} is not necessarily related.

In order to perform a comparison of different correlation techniques applied to all the variables in the 2011 Census, it is necessary to establish a baseline for reference. For this reason, the full set of 2,558 Census variables was run with the following analysis:

$$I_{ij} = \frac{1}{S_0} \sum_i \sum_j \frac{(Y_i - \mu_y)}{\sigma_y} \times W_{ij} \times \frac{(X_j - \mu_x)}{\sigma_x} \tag{7.2}$$

$$S_0 = \sum_i \sum_j W_{ij} \tag{7.3}$$

In this instance, the weight (W_{ij}) is a simple inverse distance function based on the MSOA centroids, where $W = 1$ for the case when $i = j$ and $W = \frac{1}{distance}$ for every other case. The distances are in kilometres based on the OSGB36 projection from the ONS Census boundary files, with the S_0 factor performing the normalisation of the

distance. No additional normalisation has been performed on the Y_i or X_j values as detailed in [War85] for the z_i' correlation variable. The computing time required for the additional floating point operations is significant in this case and the normalisation is unnecessary for the comparison between variables. While this distance calculation could give too much weight to the autocorrelation case and an exponential inverse distance function might also improve the results, the first task is to obtain a baseline set of data. Then any alternatives to this technique can be investigated and compared. Another possible formula for the intra-zonal case is: $\sqrt{\frac{area}{2\pi}}$, which is based on the radius of a circle within another circle of equal area where the total area is the area of the MSOA zone.

The computation of the baseline dataset was achieved using 30 processing cores to speed up the computation time and obtain a result in approximately 3 days. With 3,272,961 (C_1) correlations to be performed on 7201x7201 values (C_2) at MSOA level, this equates to a speed of 12.6 correlations per second across all parallel cores. Equation 7.4 shows the calculation of the number of correlations (C_1) based on the number of dataset variables (N), multiplied by the number of iterations for each correlation (C_2) using the number of MSOA areas (A). From the correlation formula in equation 7.2, there are two subtracts, two divides and two multiplies inside the correlation calculation, giving a total of 6 floating point operations (C_3), ignoring any load/store delays and assuming spatial weights (W) and per dataset means (μ) and standard deviations (σ) can be pre-calculated. This, in essence, is why the data takes three days to analyse.

$$
\begin{aligned}
C_1 &= \frac{(N)(N+1)}{2} && (7.4)\\
&= 3272961 \text{ N=2558} \\
C_2 &= A^2 && (7.5)\\
&= 51854401 \text{ A=7201} && (7.6)\\
C_3 &= 6\text{FP Ops} \\
C_1 \times C_2 \times C_3 &= 169,717,432,151,361 \times C_3 && (7.7)\\
&= 1,018,304,592,908,166 \text{ FP OPS}
\end{aligned}
$$

The calculation from equation 7.4 is shown graphically in figure 7.1 as the number of datasets is increased from 1 to 5000. The three lines on the graph show the time

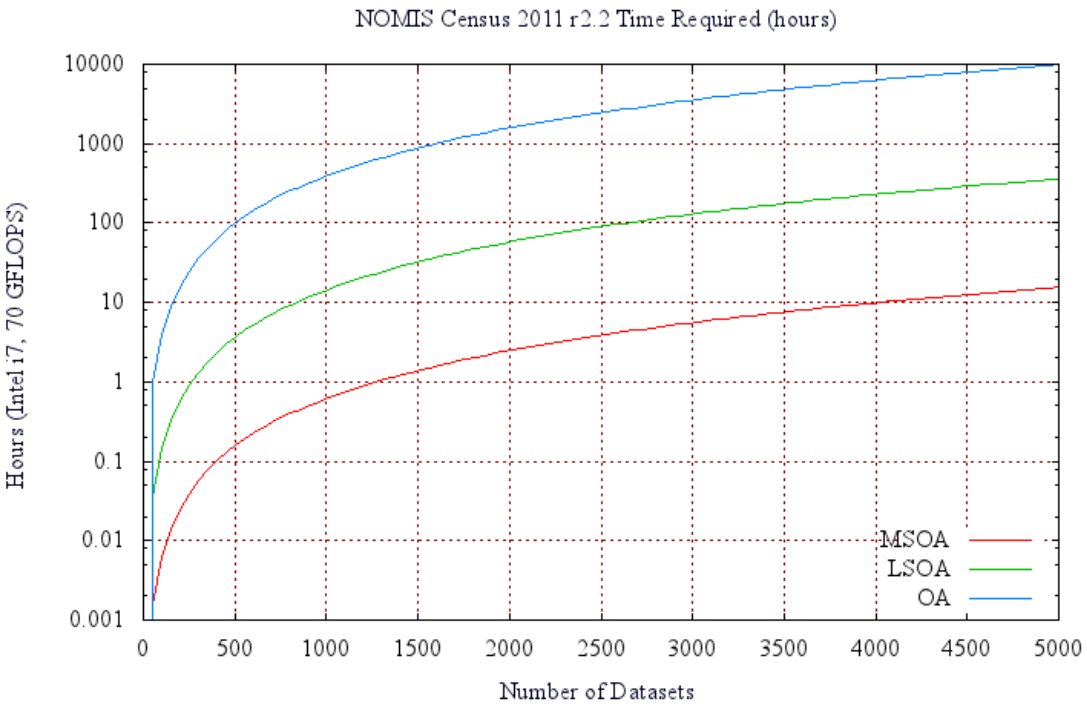
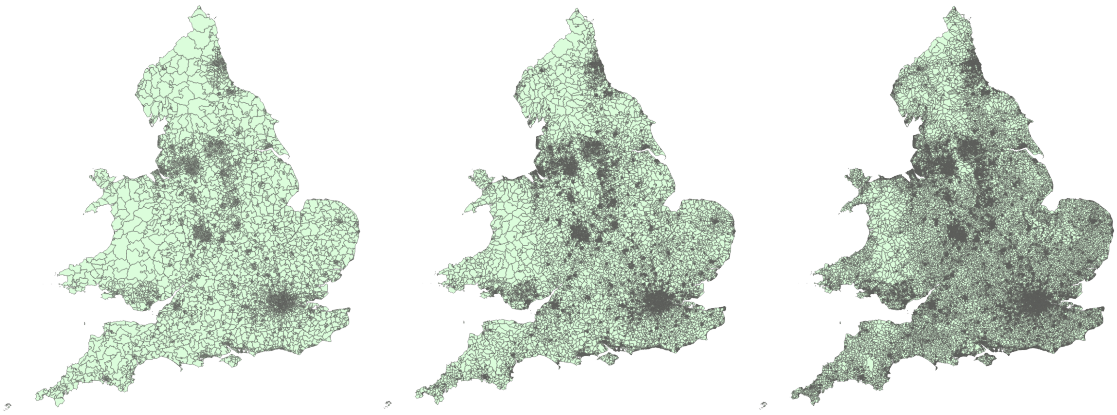


Figure 7.1: Estimated time taken in hours for an Intel i7 running at a sustained 70 GFLOPS to compute the correlation of 1 to 5000 datasets for MSOA, LSOA and OA geographies.



(a) MSOA (A=7201). (b) LSOA (A=34753). (c) OA (A=181408).

Figure 7.2: ONS Geographies for MSOA, LSOA and OA levels.

required for MSOA, LSOA and OA level data (figure 7.2). One additional piece of information is the time required to add a new dataset to the existing correlation matrix. Intuitively, this is simply the amount of time required to correlate the new map with all the existing maps to add a new row and column, so increases linearly with the number of datasets, N. Rows and columns are symmetric in the leading diagonal, so the factor is N rather than $2N$, although it could be argued that the autocorrelation case is always 1. This result is important in the case of a data store where the relations between all

the existing datasets are known and then a new piece of information is added. Even a linearly increasing time factor here will eventually lead to a point where adding a new piece of information is no longer viable due to the amount of processing time required. The only question is at what point this happens, which is dependent on the underlying geography (C_2 and C_3) and the processing speed.

Figure 7.3 shows the histogram of correlation values using equation 7.2 on every unique pair of Census variables from the 2011 R2.2 bulk release. This graph is used to determine a confidence interval for the following visualisations.

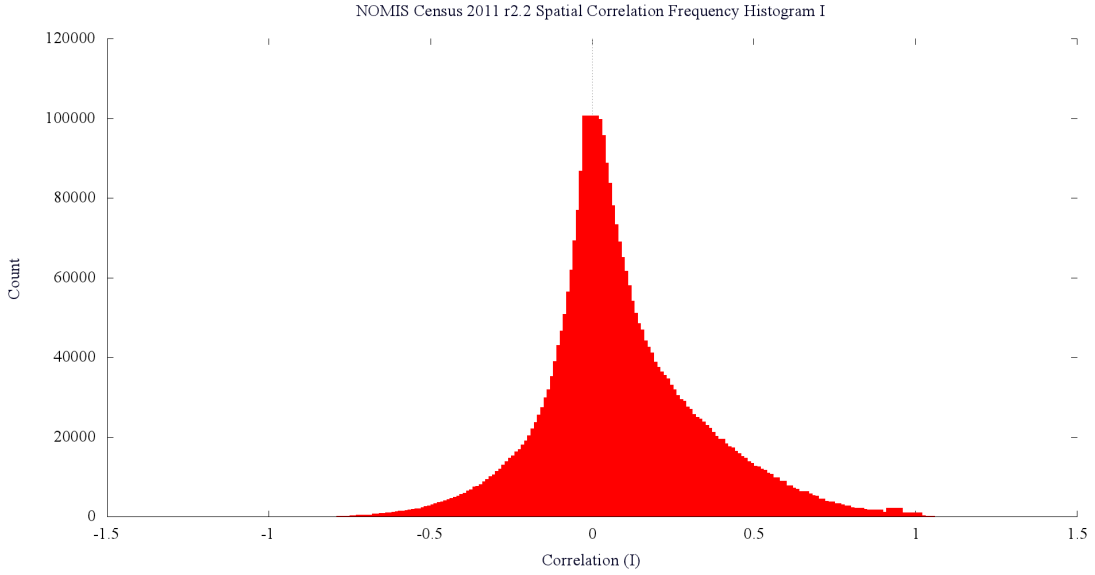


Figure 7.3: Correlation histogram of the NOMIS Census 2011 R2.2 bulk release containing 2551 variables and 3,253,355 correlation values.

One feature which is immediately noticeable from the graph is the fact that the positive side has greater area than the negative side, suggesting that positive correlations between datasets are more likely than inverse relationships. There are 2,091,817 positive values in the dataset (64%) compared to 1,161,538 (36%) out of a total of 3,253,355 samples with none identically equal to zero. From the initial calculations after cleaning and merging data from multiple runs, there should be 3,272,961 correlation values, so 19,606 (0.6%) are missing. A scan of the total number of variables processed gives a total of just 2552, so 6 are missing completely. By plotting the number of variables processed and normalising by the number in each table, the difference can be plotted. This is shown in figure 7.4 which identifies the 5 main tables affected as: *QS211, QS607, KS611, KS612 and KS613*. This allows the six variables which are missing from the data completely to be identified as: *KS611EW0016, KS611EW0031,*

KS612EW0016, KS612EW0031, KS613EW0016 and KS613EW0031. Further investigation of the Census data shows that these six variables are zero for every area, which is why the correlation test fails. The standard deviations used for the normalisation in equation 7.2 are zero, resulting in a divide by zero and a failure. Although, in strict terms, the failure is the correct result as the correlation cannot be performed with this data.

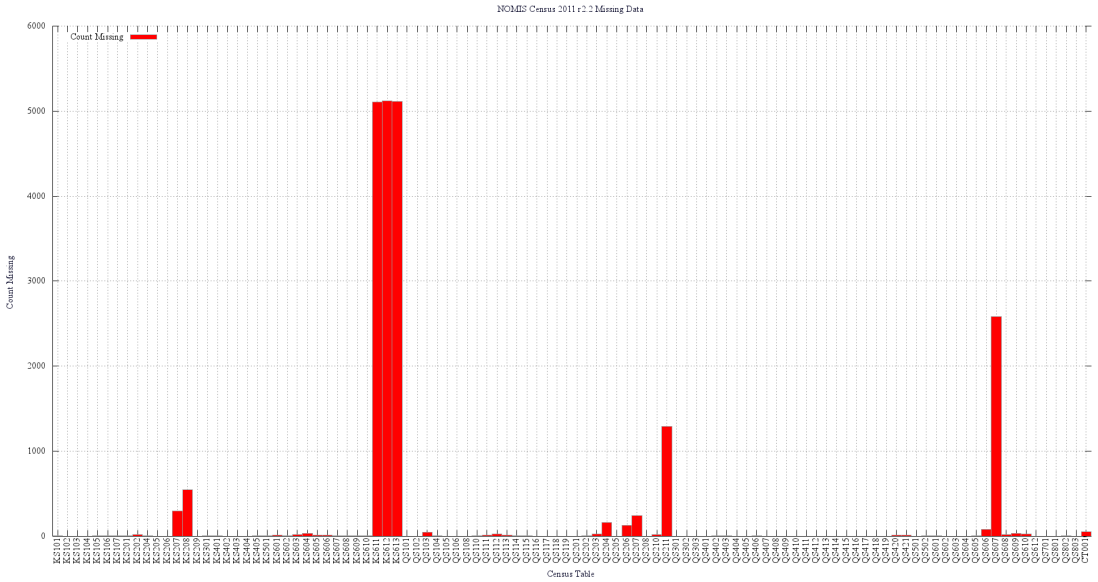


Figure 7.4: Variables missing from the NOMIS Census 2011 correlation grouped by table.

One potential first step in the analysis is a correlation matrix, as used in [WSB03] for the analysis of neurological data. There is a problem with the visualisation here as 2558x2558 pixels is bigger than most screens or pages. Full HD resolution is 1920x1080, 4K Ultra HD is 3840x2160 and 8K Ultra HD is 7680x4320, while A4 paper (8.3x11.7in) at 600 DPI is 4980x7020 dots. The correlation matrix and dendritic clustering analysis of [WSB03] was only performed on 10 variables and, using the graph from figure 7.5, only a fraction of the whole dataset is likely to be of interest as this is a data mining outlier detection task.

As the Census data is grouped into 104 tables, with each table containing between 1 and 251 variables, which are all related to the same topic, the first visualisation is an aggregate correlation matrix at the table level.

Figure 7.5 shows the table histogram counterpart of the individual variables from figure 7.3.

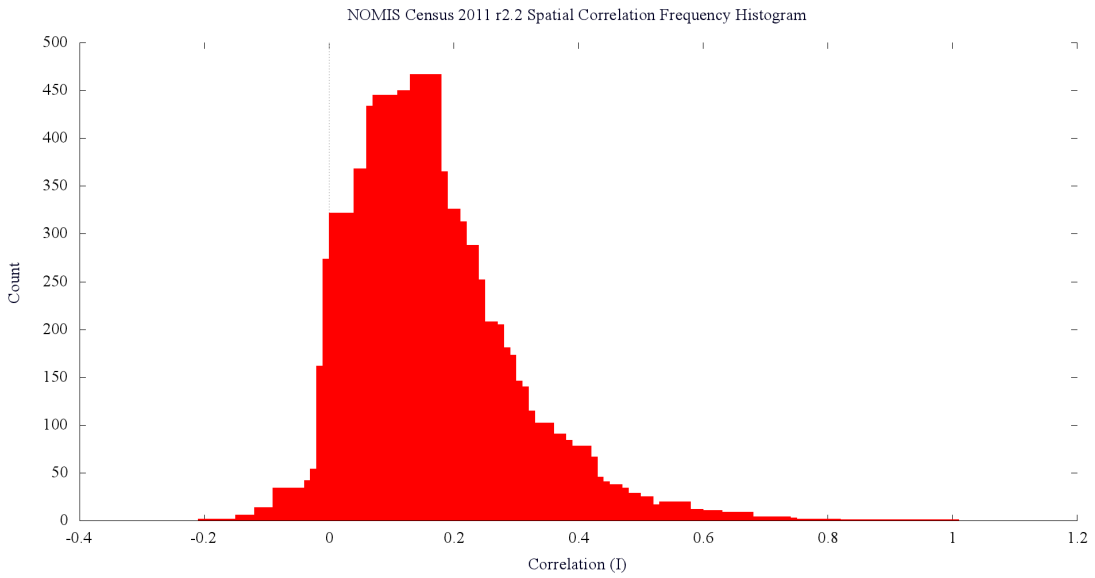


Figure 7.5: Correlation histogram of the NOMIS Census 2011 R2.2 bulk release containing an aggregate of the 104 tables, $n = 104 \times 104 = 10816$, $\mu = 0.17$, $\sigma^2 = 0.015$, $\sigma = 0.12$.

With the reduction in data it is possible to plot a correlation matrix showing how all the datasets are related. This is shown in figure 7.6, which is the initial visualisation before any row re-ordering has been performed. Even at this resolution, plotting the table names on the rows and columns would be illegible, so the only viable alternative is an interactive visualisation.

Libraries already exist which can perform this type of analysis, for example, the R Studio library called 'Complex Heatmap'[3] and the D3 'Flare' library for creating tree diagrams and dendrograms. Mike Bostock's visualisation of the 'Les Miserables' characters' relationships[4] served as the basis for the charts in figure 7.6.

The graph in figure 7.7 shows an interesting result which is visible in the repeated rows present in the previous correlation matrix. If the first three tables are plotted on the same graph with the Y axis representing the correlation coefficient for each of the Census tables plotted against the x axis, the similarity is immediately apparent.

Table 'KS101' is described as 'Usual Resident Population' and is a count of people broken down into 12 variables on the themes of sex, school children and lives in household or communal establishment. The second table, 'KS102' is a breakdown of population into 35 age ranges, while 'KS103' contains 13 variables which are counts of single, married, divorced or widowed. All three are essentially population based statistics and so have a high degree of similarity. The interesting parts of figure 7.7 are

[3] R Studio Complex Heatmap: *https://github.com/jokergoo/ComplexHeatmap*.
[4] See: *http://bost.ocks.org/mike/miserables/* for the D3 'Les Miserables' example.

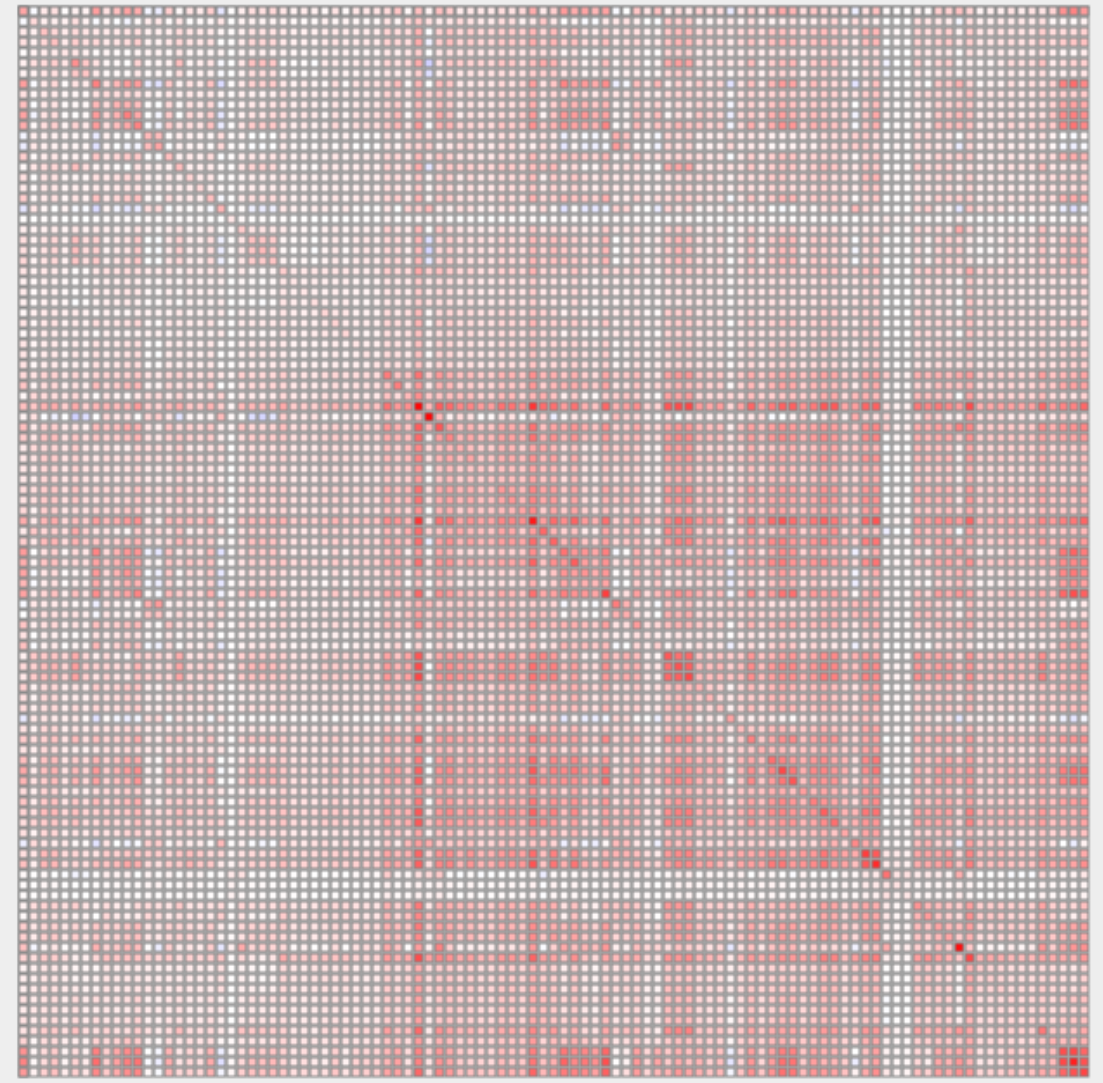


Figure 7.6: Correlation matrix for the 104 tables.

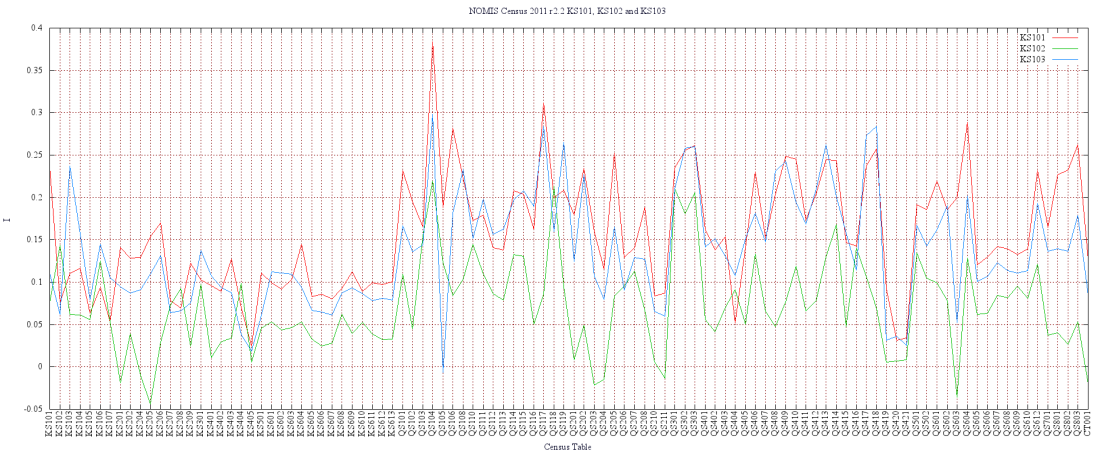


Figure 7.7: Graph of correlation coefficient (I) for first three tables (KS101-KS103) against all tables. KS101 is the 'Usual Resident Population' table, KS102 is 'Age Structure' and KS103 is 'Marital Status'.

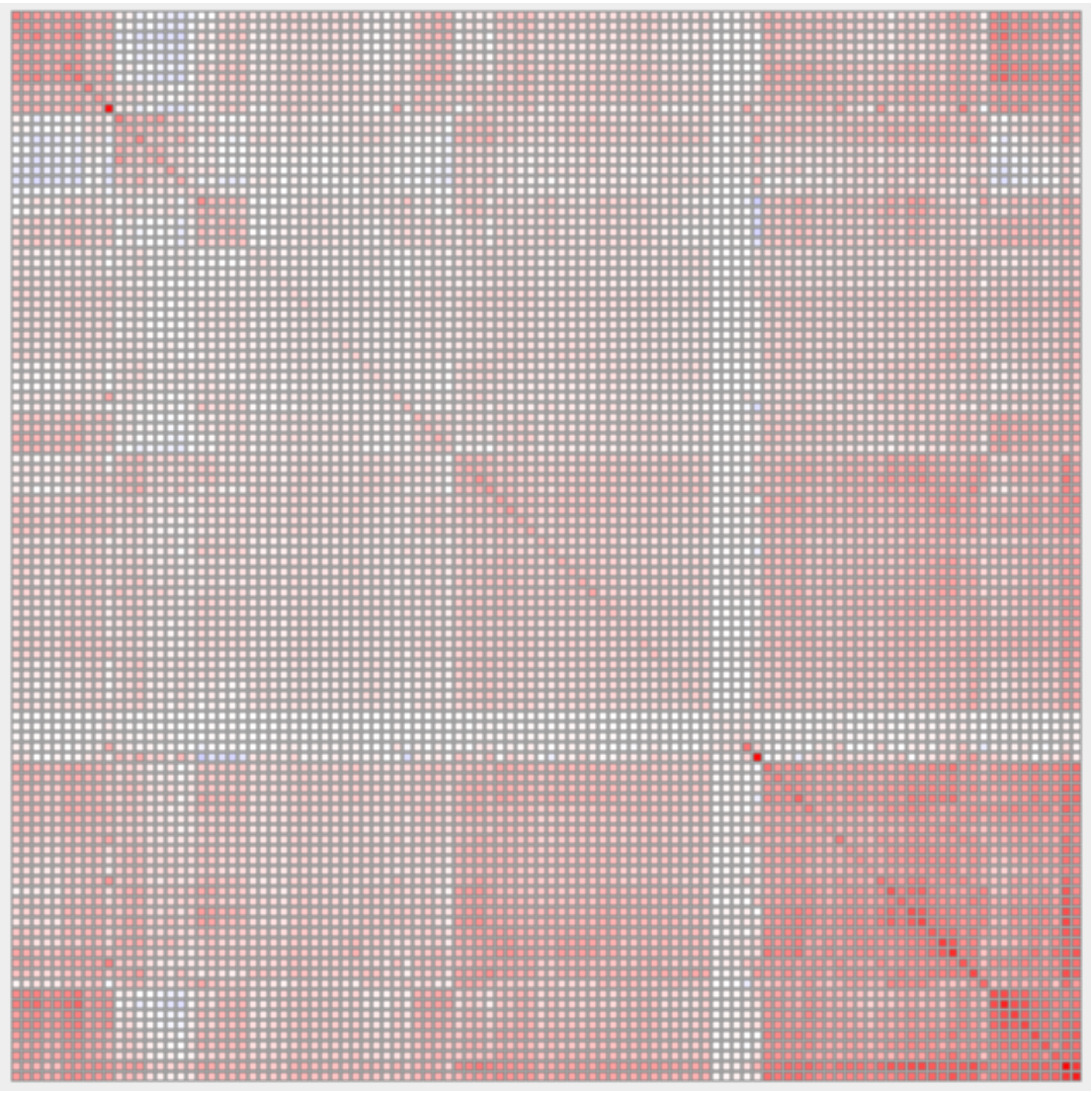


Figure 7.8: Correlation matrix for the 104 tables with the rows and columns ranked using hierarchical agglomerative clustering to highlight similar groups.

the places where they diverge, but the reason for highlighting this property is to build an ordering system for the correlation matrix which helps to identify clusters.

The correlation matrix in figure 7.8 is obtained by sorting the rows and columns of the previous figure 7.6 according to a simple hierarchical agglomerative clustering algorithm. Following the basic technique outlined in [WSB03], the dendrograph tool in [McC68] and the overview of clustering methods in [WEH11, pp273-293], the rows are sorted according to similarity. The distance measure used here is the root mean square error between clusters, following the 'centroid-linkage' method listed in [WEH11]. The initial data is taken from the rows of the correlation matrix. The data starts off as a list of 104 feature vectors, one for each Census table. These groups are labelled as G_i for $i = \{0, 1, 2, \ldots, 103\}$. These are the starting groups for the bot-

tom up clustering algorithm to begin merging. A group (G_i) contains a feature vector, $v = \{v_0, v_1, \ldots, v_{103}\}$, where v_j corresponds to the correlation value (I_{ij}) from the relevant $(row_i, column_j)$ cell of the correlation matrix. The rows are then re-ordered according to the rank obtained from the order in which the groups were merged. Many algorithms exist that are suitable for this type of analysis on a correlation matrix, for example, C4.5 and other top down inductive decision tree algorithms or clustering algorithms, so there is the potential for further research into the most appropriate in this context.

The justification for examining the correlation factors using hierarchical clustering stems from the hypothesis that the discovery of higher level structure in the data makes the visualisation of all the relationships between the data in the data store possible. The idea of investigating clusters of correlations can be explained as follows: if the existence of a "super feature" is assumed (for example, "population effect"), then this would correlate highly with all the members of the group, but, by virtue of the nature of the correlation function, these group members would likely also correlate highly with each other. It should be recognised, however, that this relationship is not guaranteed, but, given the 2,558 element vectors being compared here, it would be a highly unusual outlier for a completely unrelated dataset to enter the cluster. The data shown in figure 7.8 demonstrates this technique working on the Census variables.

In addition to displaying data as a correlation matrix, the matrix can be transformed into a directed network graph linking vertex i to vertex j by an edge weighted by the correlation value I_{ij} from the matrix.

Edge thresholding can be used to extract clusters from the initial fully-connected network, showing which data tables are most related. By a process of experimentation using the Gephi open source software[5], an edge weight threshold of 0.6 results in a single cluster of 19 connected tables as shown in figure 7.9.

The reasoning behind this method is to determine structure, but the highly correlated core of the Census serves to hide the more interesting structure beneath. Taking the data from figure 7.9 as an example, *QS204* is a count of all people by sex, so is highly correlated with most of the other Census tables which are all based on counts of sub-populations. *QS205*, English language proficiency and *QS802* and *QS803*, age of arrival in the UK and length of time spent in the UK are also related. Other factors, like

[5]Gephi: *http://gephi.github.io/*.

country of origin, main language and religion are also related for obvious reasons, so the task becomes one of mining the data to remove these trivial relationships, leaving only the interesting outlier cases which need to be explained.

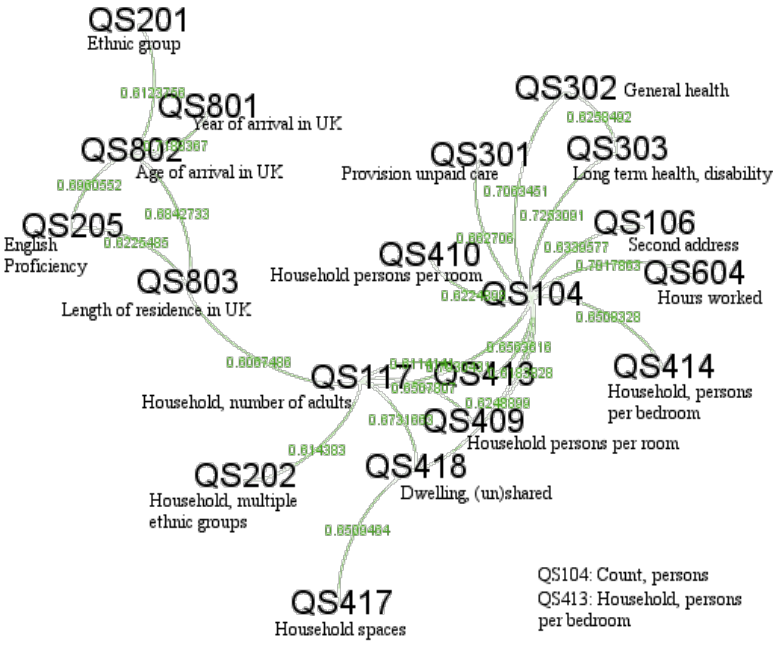


Figure 7.9: Network graph for the 104 tables with edge weights representing the correlation between tables. This plot was produced using the Gephi program, using a filter to display only connections with edge weight greater than 0.6.

7.1.2 Spatial Indexing, Distances and Neighbours

Having analysed the Census 2011 data, the next step is to explore alternative algorithms which can produce similar results in a fraction of the time. The ultimate aim is to find a method using 'Stream Mining' [WEH11, pp380], or 'Frugal' algorithms, where each data set is loaded just once and an estimate of similarity is obtained for all the data in a single pass.

Various different methods for choosing weights have been proposed, which all have a bearing on the number of calculations that need to be performed:

- The weight for any two different locations is constant
- All observations within a defined distance (radius) have a constant weight
- k-nearest neighbours (KNN) have fixed weight, all others are zero
- Weights are proportional to distance (d): $W = \frac{1}{d}$, $W = \frac{1}{d^2}$ or only up to a cut-off distance
- Adjacent areas have constant weight, all others are zero
- Pick a random subset of the data

The six weighting schemes listed here are included to show a variety of methods can be tested using the software presented in this thesis. Each has relative strengths and weaknesses and there is no "right answer" as to which one out-performs the rest on any task, so the first requirement for a generally useful spatial cross correlation algorithm is to give researchers the ability to experiment with all of the different options. However, in the interests of proving the performance and results of using the system on real data, at least one of these needs to be tested and compared against the computationally intensive full spatial cross correlation algorithm. With the first option where, "the weight for any two different locations is constant", this came from the "Geoda" handbook[6] where binary weights were used to indicate a neighbourhood or "closeness" relationship without having to express distance. While topological relationships are a factor of interest, this does not satisfy the requirement of limiting the geographic extent for performance, or providing an equivalence test with the cross correlation data previously calculated. Moving on to the next option, which is "all observations within a radius have constant weight", this is similar to the next option of taking the "k-nearest

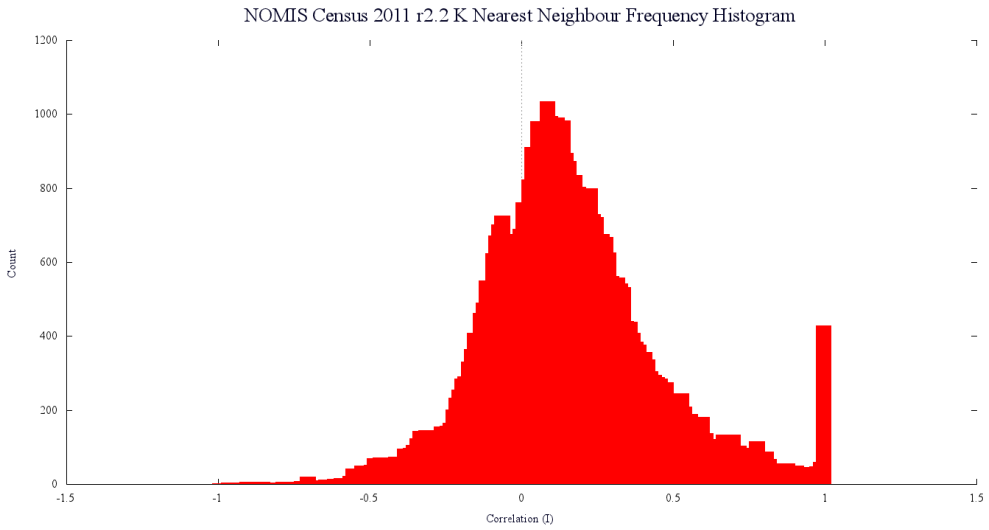
[6] Geoda: *https://geodatacenter.github.io/workbook/4a_contig_weights/lab4a.html*.

neighbours" and the penultimate option of "adjacent areas have constant weight". Of these, "k-nearest neighbours" would seem to provide the closest match to the existing cross correlation data as the other two are topological. The "weights proportional to distance" is the method already used for the test, while "pick a random subset" will require excessive sampling to provide a quick test and fails on performance grounds. This leaves the 'k-nearest neighbour' test as the leading candidate for a comparative test of quality and performance against the existing control data.

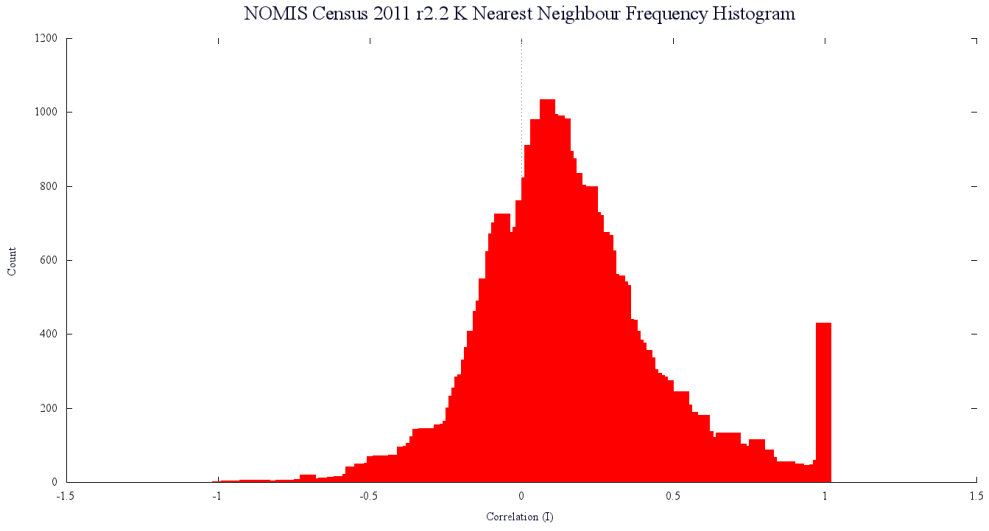
Where weights are zero, no calculations need to be performed, so indexing the data spatially and using this to find neighbours will improve performance. The 'k-nearest neighbours' scheme relies on sorted distance, while the 'adjacent areas' method is based on topology, but both can be pre-calculated. Finally, 'pick a random subset of the data' might seem to be a strange technique, but its roots come from data mining, where random folds of data are used in training a classification system [WEH11].

Performing correlations between areas separated by large distances only adds a small amount to the overall correlation factor, so one possible heuristic is to limit the distance and reduce the number of operations. This requires a fast way of finding areas within range of the test area, which is achievable using spatial indexing. As an alternative to spatial distance, the topology can also be used, with only adjacent neighbours used for the match, or extended to 2-neighbours (the neighbour of my neighbour) and successively remote links. By pre-calculating the topology as a network graph this technique can be applied to the MSOA baseline data. A similar technique is discussed in [Mul75] for finding associations in choropleth maps using a network graph matching method.

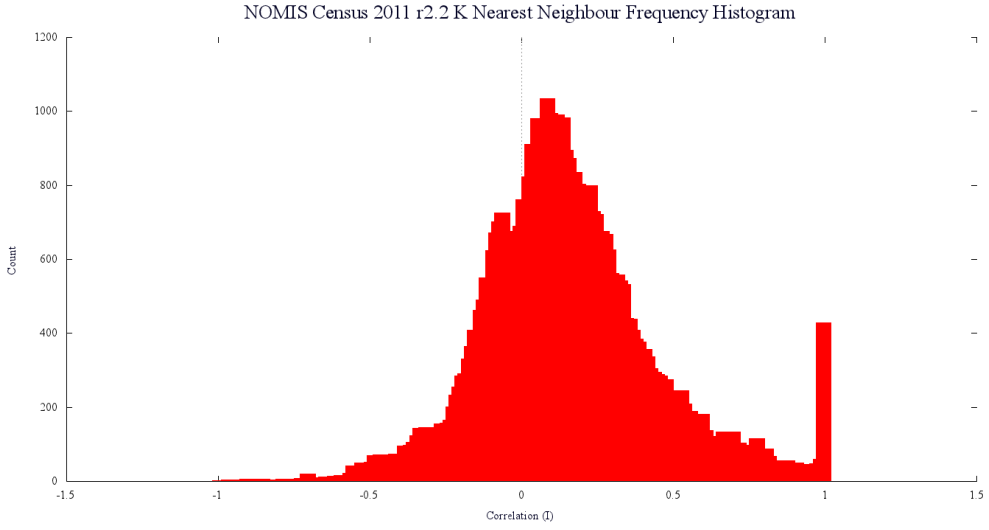
Figures 7.10a, 7.10b and 7.10c show the results of running the k-nearest neighbour algorithm on the NOMIS Census 2011 data using $k = 3$, $k = 4$ and $k = 5$ respectively. As k is increased, the amount of time taken increases linearly, so this technique is useful for small k, tending towards the full spatial cross correlation time as $k \longrightarrow 7200$. The nature of the algorithm is such that, as k is increased, areas further away are added into the equation, thus increasing the precision by a factor that diminishes by distance with increasing k. The k-nearest neighbour value tends towards the full spatial cross correlation value, but by ever reducing amounts. This is borne out in the following analysis of the data, all of which is available for download at the following address: *http://www.maptube.org/PhDData*.



(a) K=3.



(b) K=4.



(c) K=5.

Figure 7.10: Frequency histogram for NOMIS 2011 Census data using KNN algorithm with number of neighbours K=3, 4 and 5. All three graphs have mean $\mu = 0.248$, standard deviation $\sigma = 0.248$ and number of samples $n = 3272955$.

The graph in figure 7.11 shows the timings for $k = 3$ to $k = 9$. The $k = 9$ data was the longest to compute and took $2842644ms$, or 47.7 minutes, which is significantly shorter than the full spatial cross correlation.

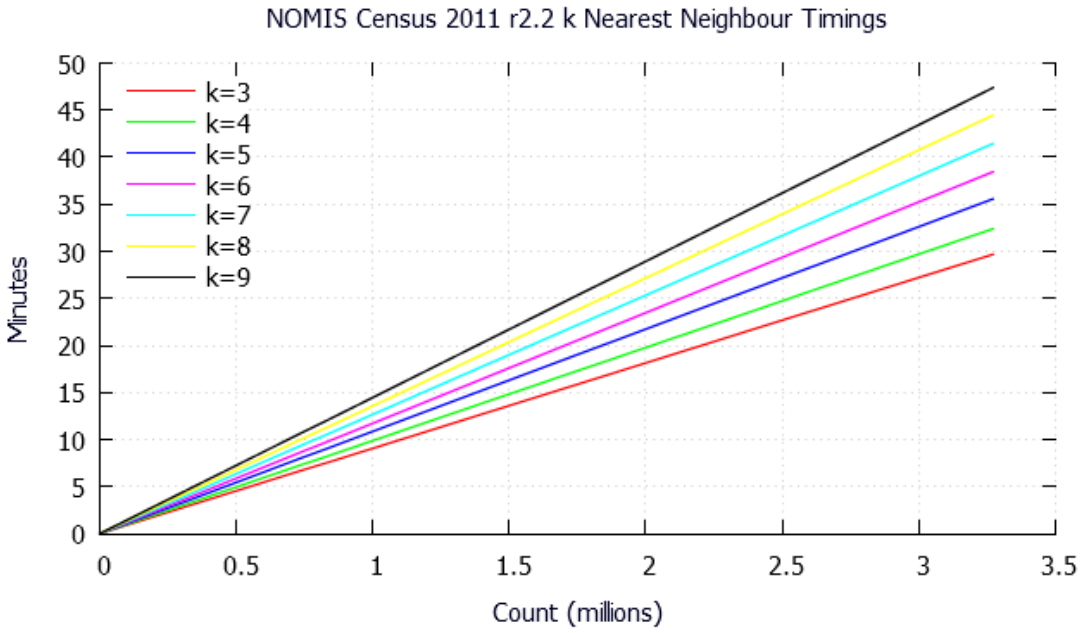


Figure 7.11: Timings for k-nearest neighbour computation for k=3 to k=9. Timings are in minutes on the y-axis.

The most remarkable feature of the data in figure 7.11 is the regularity of the lines. These are not straight lines plotted between two points. Each one of the $k = 3$ to $k = 9$ gradients shows the cumulative time to compute the 3.2 million correlations required, so the x-axis is composed of 3.2 million line segments connected together. If the processing slowed down part way through the run, then this would show as a bump in the line. The timing figures from these experiments demonstrate the speed and scalability of this approach, so the remainder of this section now looks into the data in detail.

The remaining question is whether the data is comparable to the original spatial cross correlation which took several days to compute? The quantity of data produced is too large for most commercial and non-commercial packages to handle, comprising 3,272,955 rows or correlation pairs. The charts produced here were created using gnuplot[7] as few spreadsheet packages can handle the row count. The WEKA[8] data mining package is able to handle data of this size, as many of the functions are able to use

[7]For more information about gnuplot the project web page at: *http://www.gnuplot.info*.

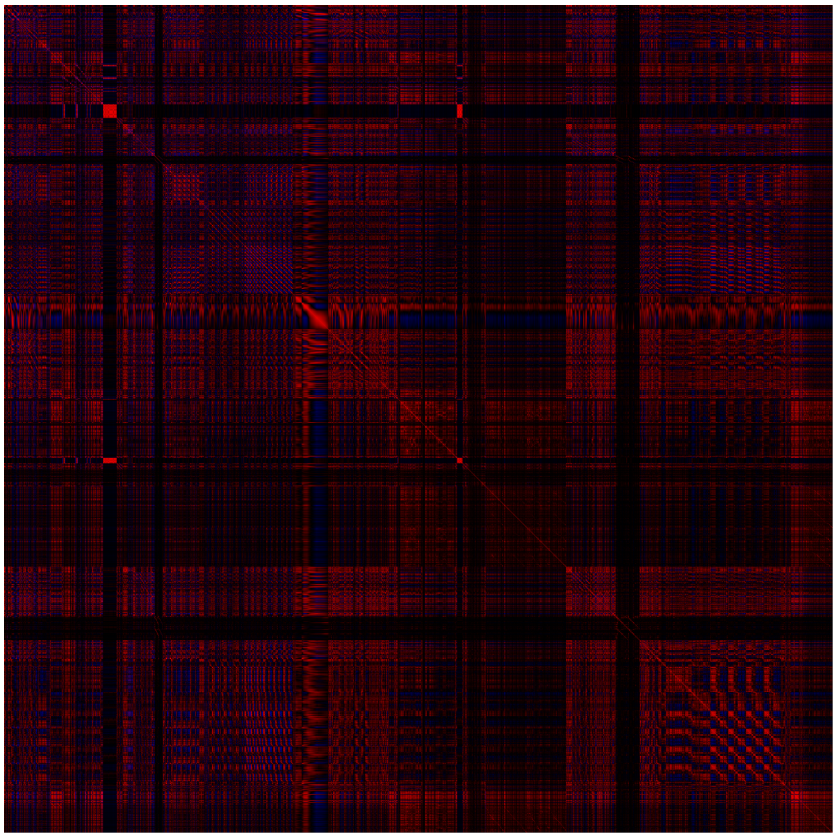
[8]WEKA is the university of Waikato's open-source data mining software downloadable from: *http://www.cs.waikato.ac.nz/ml/weka*. For more information see: [Hal+09].

samples from a random subset of the data, although the percentage of the subset size can be configured from 0% up to 100% of the data. In this instance, only the upper and lower tails of the data are of interest, which was the original justification for using a more efficient algorithm rather than cloud computing in the first instance.

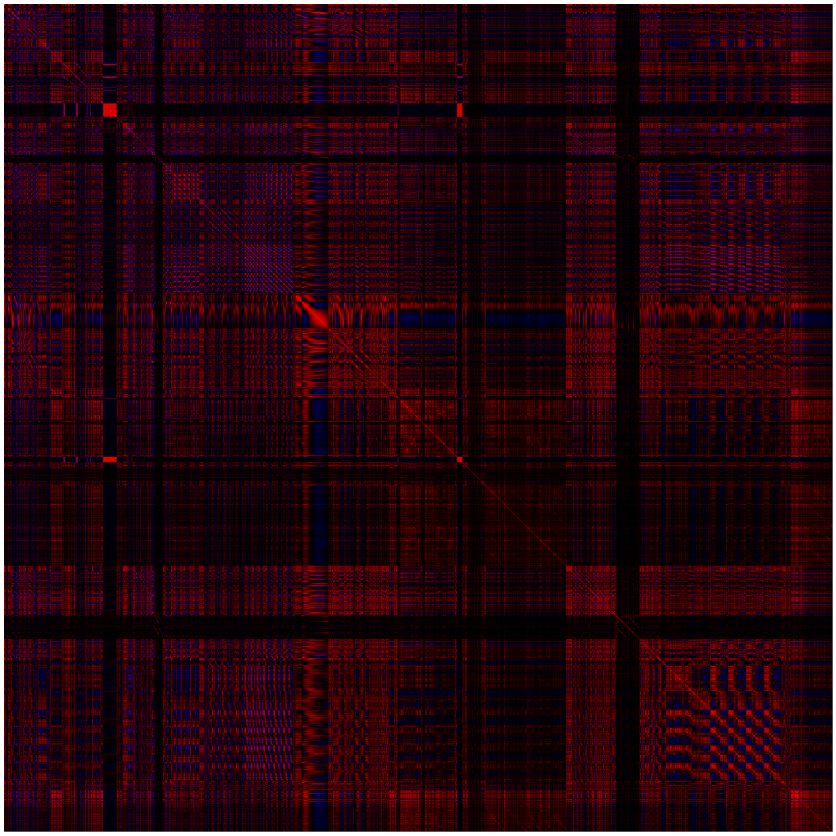
As this is an outlier detection case, a large part of the data can be discarded, leaving only the upper and lower tails. The correlation measure is only a heuristic to find these outliers and any other possible links in the data. By setting a threshold of 0.6 so only the upper tail, positive correlations, are left, the spatial correlation data can be compared against the k-nearest neighbour data. Using a correlation matrix built from the raw correlation pairs of variables, which is symmetric in the leading diagonal, a network graph can be built where nodes are the variables and edges between variables are weighted by the magnitude of the correlation between the pair. When this transformation is applied to the data, the three k-nearest neighbour datasets all have 2558 nodes and 126,196 edges. The full spatial correlation data results in a slightly different graph, with 2552 nodes and 229,773 edges. This is a consequence of unequal scaling between the two different types of data, with the 0.6 threshold selecting more of the spatial correlation sample. For the purposes of the following comparison, though, this does not affect the result. The comparison is then performed between the original spatial correlation data and the k-nearest neighbour data with $k = 5$, working on the assumption that the $k = 3$ and $k = 4$ data are similar.[9]

The comparison between the full spatial cross correlation and the $k = 9$ k-nearest neighbour is shown in figures 7.12a and 7.12b. These figures are a plot of the raw data which gives a visual impression of the data under analysis. Each figure contains 2,558 pixels by 2,558 pixels, one for the correlation data for every possible pair of Census variables. The colour scheme is chosen so that a correlation of 1.0 between variables gives a full red pixel, -1.0 gives full blue and zero gives black. In between are interpolated shades of red and blue depending on positive or negative. In this case, the data is predominantly positive and strong negative correlations are rare. The top left to bottom right diagonal displays the self-correlation values of 1.0, with symmetry of the data either side of this line, as the correlation formula and k-nearest neighbour formulas are both commutative. Visually, both matrix plots show data with a high degree of similarity.

[9]The probability density functions in figures 7.10a, 7.10b and 7.10c and a correlation test between the datasets does confirm this to be true.



(a) Spatial Correlation.



(b) KNN K=9.

Figure 7.12: Correlation matrix for spatial cross correlation compared to k-nearest neighbour with k=9. Both matrices show a high degree of similarity, RMS error is 0.0138. Full red pixels indicate a correlation of 1.0, black 0.0 and full blue -1.0. The maps being compared are ordered along the rows and columns, so the pixel at row 0, column 10 is map 0 correlated against map 10. Note the correlation of 1.0 along the 45 degree axis and symmetry about this line. NOTE: the images for this figure can be viewed at full scale online at: *http://www.maptube.org/PhDData*.

Starting with the correlation data in table 7.1, there is a high degree of similarity between these correlation matrices. This is to be expected as this is an outlier detection case, where only a small percentage of the data is important and hidden in amongst a large number of trivial relationships. Also shown in table 7.1 is the compute time for the k-nearest neighbours, which varies from 29.7 minutes for k=3 up to 47.4 minutes for k=9, but with only a 0.000018 improvement in RMS error compared to the longer running spatial cross correlation algorithm, which takes over 24 hours to compute.

KNN	RMS Error	Compute Time (ms)
3	0.01376536	1780365
4	0.01377724	1943541
5	0.01377787	2133845
6	0.01376492	2306365
7	0.01375999	2486563
8	0.01377046	2667138
9	0.01375987	2842644

Table 7.1: RMS error between spatial cross correlation and k-nearest neighbour. k=9 is the minimum error as expected, but k=7 is a close second.

Metric	k=3	k=4	k=5	k=6	k=7	k=8	k=9
RMS	0.01377	0.01378	0.01378	0.01376	0.01376	0.01377	0.01376
Matrix Correlate	0.99879	0.99879	0.99879	0.99879	0.99879	0.99879	0.99879
Sorenson-Dice Index	0.97531	0.97530	0.97530	0.97531	0.97531	0.97531	0.97531
Jaccard Index (intersection over union)	0.91833	0.91832	0.91832	0.91833	0.91833	0.91833	0.91833
Sum Of Ratios	6222399	6222387	6222383	6222403	6222406	6222396	6222408
Average Ratio	0.9509	0.9509	0.9509	0.9509	0.9509	0.9509	0.9509
Absolute Difference	51844.77	51848.39	51849.04	51844.22	51843.15	51845.93	51842.96
Average Absolute Difference	0.007923	0.007924	0.007924	0.007923	0.007923	0.007923	0.007923
Absolute Percent Difference	-296960.1	-296960.5	-296960.5	-296959.8	-296959.5	-296959.8	-296959.4
Average Absolute Percent Difference	-0.04538	-0.04538	-0.04538	-0.04538	-0.04538	-0.04538	-0.04538
Information Difference	0.000651	0.000653	0.000654	0.000650	0.000649	0.000651	0.000649

Table 7.2: Correlation statistics for each k-nearest neighbour matrix for k=3 to k=9, compared with the spatial cross correlation matrix.

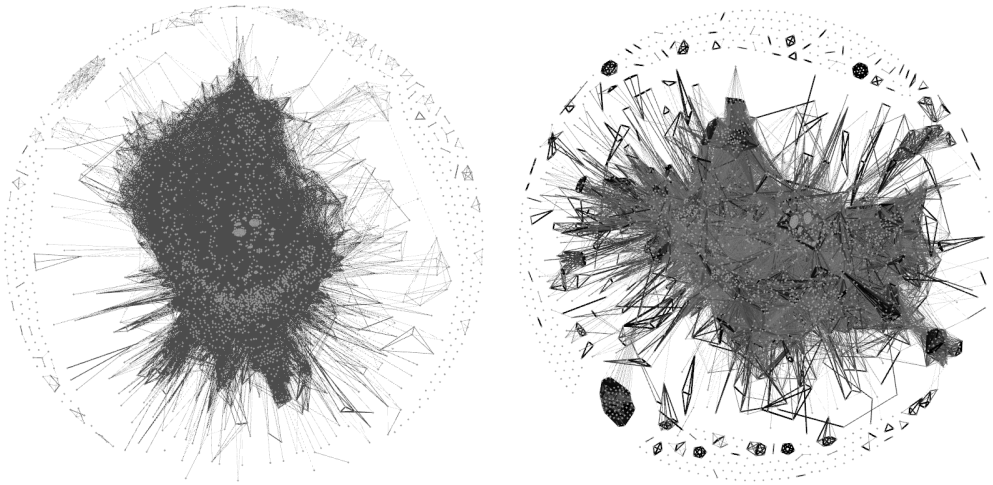
The formulas used to calculate the similarity metrics in table 7.2 are listed in the following table 7.3.

While the RMS error for the two matrices shows a 1.4% error (0.0138 from 7.2) on 3.2 million sample values, as the nature of the data is outlier detection, it is important to show that the structure of the two network graphs is equivalent. In this application,

Metric	Formula
RMS	$\theta = \sqrt{\frac{\sum_{ij}(X_{ij}-Y_{ij})^2}{n(X)}}$
Matrix Correlate	$r = \frac{\sum_{ij}(X_{ij}-\bar{X})(Y_{ij}-\bar{Y})}{\sqrt{\sum_{ij}(X_{ij}^2-\bar{X})}\sqrt{\sum_{ij}(Y_{ij}^2-\bar{Y})}}$
Sorenson-Dice Index	$\phi = \sum_{ij} \frac{min(X_{ij},Y_{ij})}{\sum_{ij} Y_{ij}}$
Jaccard Index	$J = \frac{\sum_{ij} min(X_{ij},Y_{ij})}{\sum_{ij} max(X_{ij},Y_{ij})}$
Sum Of Ratios	$\psi = \sum_{ij} \frac{X_{ij}}{Y_{ij}}$
Average Ratio	$\bar{\psi} = \frac{\psi}{n(X)}$
Absolute Difference	$G = \sum_{ij} \lvert X_{ij} - Y_{ij} \rvert$
Average Absolute Difference	$\bar{G} = \frac{G}{n(X)}$
Absolute Percent Difference	$H = \sum_{ij} \frac{\lvert X_{ij}-Y_{ij} \rvert}{X_{ij}}$
Average Absolute Percent Difference	$\bar{H} = \frac{H}{n(X)}$
Information Difference	$p_{ijx} = \frac{X_{ij}}{\sum_{ij} X_{ij}}, p_{ijy} = \frac{Y_{ij}}{\sum_{ij} Y_{ij}}$
	$I = \sum_{ij} p_{ijx} \log_{10}(p_{ijx}/p_{ijy})$

Table 7.3: Matrix correlation formulas used to compare the k-nearest neighbour matrices with the spatial cross correlation matrix. Where X and Y are the two matrices being compared, which are indexed by the i and j indices. The term, $n(X)$, is used to denote the number of elements in matrix X.

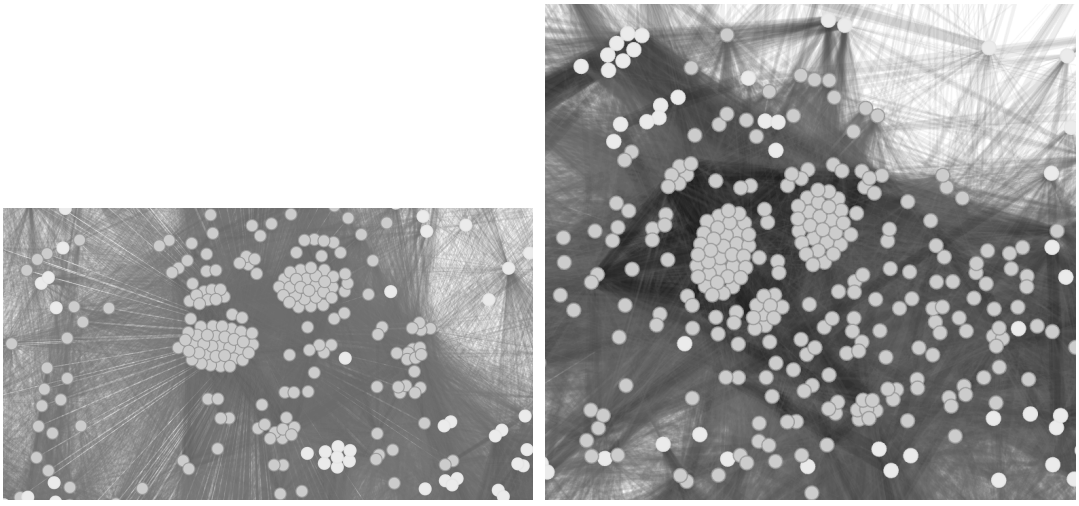
the correlation coefficient is bounded to the interval $[-1.0, +1.0]$, which prevents this from being a scale-free network. It also posses a highly connected core structure. For this reason, the identification of features in both datasets is taken as an indicator of similarity. Figures 7.13a and 7.13b show the overview plots of all the data using the upper tail threshold of $I \geq 0.6$. The layout scheme used is the Fruchterman Reingold layout, $F_a = \frac{d^2}{k}$ and $F_r = \frac{k^2}{d}$, where F_a and F_r are the attraction and repulsion forces respectively, d is distance and k is the spring constant.



(a) Spatial Correlation. (b) KNN k=5.

Figure 7.13: Overview of correlated variables from the 2001 Census showing graphs from the spatial correlation and k-nearest neighbours methods (threshold $I \geq 0.6$).

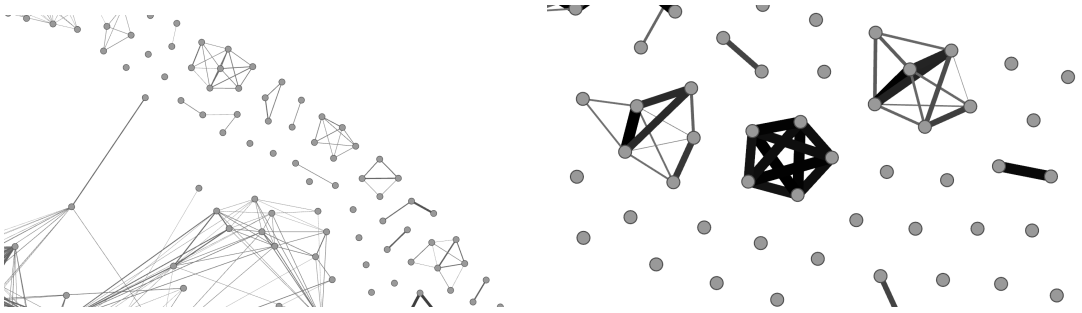
Although visually different, the first feature which is apparent is the central core present in both sets of data. This is shown in more detail in figures 7.14a and 7.14b which show a core of variables which are fully connected and forming two distinct clusters.



(a) Spatial Correlation. (b) KNN k=5.

Figure 7.14: Overview of correlated variables from the 2001 Census showing graphs from the spatial correlation and k-nearest neighbours methods (threshold $I \geq 0.6$). The view shows the fully connected central core of people counts from every table.

Further investigation of these two groups reveals that they are predominantly the first variable from each table (e.g. KS102EW001, KS103EW001). Every Census table contains the total count of population for each MSOA area, so, obviously, every first



(a) Spatial Correlation. (b) KNN K=5.

Figure 7.15: Overview of correlated variables from the 2001 Census showing graphs from the spatial correlation and k-nearest neighbours methods (threshold $I \geq 0.6$). The view shows the 'Mining and Quarrying' cluster.

variable in every table forms a cluster. The interesting part of this data is where a variable other than 'KSxxxEW001' breaks into the group. Some of the variables from table 'KS202' on national identity appear in this cluster, including 'English', 'Scottish', 'Irish' and 'Welsh'.

Finally, picking an isolated cluster of 5 fully connected nodes which can be identified in both datasets, the counts of people working in the sector 'Mining and Quarrying' demonstrate the strong correspondence between the two methods. Figures 7.15a and 7.15b show the same cluster of 5 fully connected clusters in each dataset. The other group of 6 visible in figure 7.15b is comprised of variables related to 'care', 'nursing' and 'communal living'.

While figures 7.15a and 7.15b show the structure of the two graphs visually, the aim is to measure the difference in structure empirically. One method for achieving this is through the average degree. The degree of a graph node is defined as its number of links[10], so average degree is the total number of links divided by the total number of nodes. In order to compare network structure, both graphs are progressively cut between -1.0 and +1.0, removing all links with weights below the cut threshold. Where the graph is represented as a matrix of connections, this can be computed as follows:

$$\bar{d} = \frac{\sum_{ij} n(M_{ij} \geq ncut)}{n(nodes)}$$

The results of running this analysis on the k-nearest neighbour data and spatial cross correlation matrix are shown in table 7.4.

[10]Here, the graphs are symmetric, so whether the degree measures in links, out links, or total links is immaterial.

ncut	k=3	k=4	k=5	k=6	k=7	k=8	k=9	SC
-1.0	2558	2558	2558	2558	2558	2558	2558	2558
-0.9	2557.887	2557.887	2557.887	2557.887	2557.887	2557.887	2557.887	2557.862
-0.8	2557.653	2557.653	2557.653	2557.653	2557.653	2557.653	2557.653	2557.64
-0.7	2556.21	2556.21	2556.21	2556.21	2556.21	2556.21	2556.21	2556.305
-0.6	2550.344	2550.344	2550.344	2550.344	2550.344	2550.344	2550.344	2550.837
-0.5	2536.429	2536.429	2536.429	2536.429	2536.429	2536.429	2536.429	2537.51
-0.4	2506.798	2506.798	2506.798	2506.798	2506.798	2506.798	2506.798	2508.572
-0.3	2449.984	2449.984	2449.984	2449.984	2449.984	2449.984	2449.984	2452.71
-0.2	2344.116	2344.116	2344.116	2344.116	2344.116	2344.116	2344.116	2347.925
-0.1	2145.504	2145.504	2145.504	2145.504	2145.504	2145.504	2145.504	2149.724
0.0	1642.823	1642.819	1642.818	1642.826	1642.827	1642.823	1642.828	1640.725
0.1	1047.44	1047.436	1047.435	1047.443	1047.444	1047.44	1047.444	1035.821
0.2	704.9727	704.9703	704.9703	704.9727	704.973	704.9722	704.973	690.6853
0.3	469.3401	469.3393	469.3393	469.3401	469.3405	469.3397	469.3405	452.7486
0.4	299.6974	299.6966	299.6966	299.6974	299.6978	299.697	299.6978	282.7299
0.5	180.0367	180.036	180.036	180.0367	180.0371	180.0364	180.0371	165.6618
0.6	99.6638	99.66302	99.66302	99.6638	99.66419	99.66341	99.66419	88.96599
0.7	51.30649	51.30571	51.30571	51.30649	51.30688	51.3061	51.30688	43.57428
0.8	24.17905	24.17826	24.17826	24.17905	24.17944	24.17866	24.17944	19.23651
0.9	9.209538	9.208756	9.208756	9.209538	9.209929	9.209147	9.209929	6.982408
1.0	0	0	0	0	0	0	0	0

Table 7.4: Average degree statistics for each k-nearest neighbour matrix for k=3 to k=9, compared with the spatial cross correlation matrix (SC). The weight cut threshold (ncut) is varied from -1.0 to +1.0 and the average degree plotted for all out links which greater than or equal to the ncut.

The average degree statistics show a high degree of similarity between the k-nearest neighbour graphs, while the difference between the k-nearest neighbour and spatial cross correlation (SC) graph is greater. This is showing the spatial cross correlation graph to have fewer connections, with the network breaking down more quickly as the threshold increases. Using the earlier data from table 7.3, the RMS error for k=9 was 0.0138, while the difference between k=9 and the spatial cross correlation matrix with a cut threshold of 0.9 is $(9.21 - 6.98)/6.98 = 0.32$. This is at the extreme end of the comparison, though, as at a cut level of 0.0, the error is $(1642.83 - 1640.73)/1640.73 = 0.0013$. Between those two points, the difference between the structure of the two networks gradually increases as the networks are split into smaller clusters.

Similarly, by applying a cut threshold to the links and analysing the number of groups formed as the increasing threshold forces the network to break down into clusters, the k-nearest data can be compared with the structure of the spatial cross correlation. The results are shown in table 7.5.

ncut	k=3	k=4	k=5	k=6	k=7	k=8	k=9	SC
0	1	1	1	1	1	1	1	1
0.1	14	14	14	14	14	14	14	14
0.2	46	46	46	46	46	46	46	50
0.3	82	82	82	82	82	82	82	85
0.4	152	152	152	152	152	152	152	165
0.5	218	218	218	218	218	218	218	230
0.6	301	301	301	301	301	301	301	320
0.7	438	438	438	438	438	438	438	464
0.8	596	596	596	596	596	596	596	661
0.9	963	963	963	963	963	963	963	1157

Table 7.5: Cluster sizes for each k-nearest neighbour matrix for k=3 to k=9, compared with the spatial cross correlation matrix (SC). The weight cut is varied between 0.0 and 0.9, with the numbers in the table showing the count of individual (disconnected) clusters formed as the links below the cut threshold are removed. The data shows the breakdown of the 10 network graphs as a comparison of how their structure is similar. The data shows that k=3 to k=9 are identical at this level of detail, while differing in structure from the full spatial cross correlation graph.

The data in tables 7.4 and 7.5 both show how the differences between the two methods for creating the network graphs from the links between maps increase as the correlation value approaches 1.0. Using the matrix correlation data in table 7.2 in isolation, this bias towards lower correlation values at the higher end of the scale is not apparent. This shows that the full spatial cross correlation is assigning higher values to pairs of maps where they are connected by correlated data further away, which is being missed by the distance cut-off heuristic which improves the speed of the k-nearest neighbour method. There is an interesting parallel to make here between using a threshold cut-off on network edge weights as a metric to analyse a heuristic which uses a distance cut-off threshold to speed up the computation of linked maps. To re-state the original hypothesis, this is an outlier detection case, where only a small percentage of the data is important and hidden in amongst a large number of trivial relationships. The fact that similar features can be extracted from both gives weight to the idea that the faster technique can be used as a heuristic to eliminate large parts of the data prior to using a more computationally intense algorithm.

7.1.3 Not the Census

Returning to the original justification for analysing the relationships between sources of data, the requirement was to be able to define structure for the vast quantities of data now being collected in data stores. The earlier quote was how to avoid data stores being just 'data dumps', so, while the example presented here has concentrated on the 2011 Census, the technique needs to be applicable to data stores generally. This is where the problem of geography becomes an issue, because all of the Census data was using the MSOA geography. The MapTube website contains data that uses points, custom geometries from shapefiles, various ONS geographies, Parliamentary Constituencies, postcodes and countries of the world.

One possible solution is to build a converter to go from one geometry to another, while another approach is to impose a new regular grid structure on the data as used in the 'gridded 24 hour population models' work of [MCL10] and the discrete geodesic grids of [SWK03]. In practice, both methods require the conversion of one geometry into another, either the geometry of one of the data sets or an intermediate grid representation. This requires a method of attributing data from the original source geometry to the target geometry. There are two ways of achieving this: build a third geometry based on the intersection of the two geometries to be compared, or stick with the target geometry as it is and attribute data values from the other. The option of building a third geometry would give a greater spatial resolution, imagine two circles that overlap slightly in the middle, where three areas would be created where there were only two in the original data. However this is attempted, though, the attributes need to be calculated for the new areas and a significant amount of geometric manipulation will need to take place. Contrast this with the gridding idea, which has already been used successfully and is well documented in the previously referenced publications. If the size of the grid cell is configurable by the user, then the effects of this process on the data could be explored by the researcher by varying the size. Also, with attribute data from the two input geometries sub-sampled to an intermediate grid that is smaller than the originals, then the boundaries of zones from both of the input geometries should be visible in the final grid. This second option has been attempted with both the MapTube website and the London Data Store. Keyword only correlations have also been made on the *data.gov.uk* site as an aid to understanding the content, with the results presented at the

National Centre for Research Methods Conference 2012[11]. The reason for performing a keyword only search here is due to the lack of availability of any geographical data for the vast majority of the datasets. The graph obtained from this experiment is shown in figure 7.16, containing all datasets connected by the 'climate' keyword. Clusters which could be labelled as 'air quality', 'pollution', 'precipitation', 'flooding' and 'transport' can be readily identified.

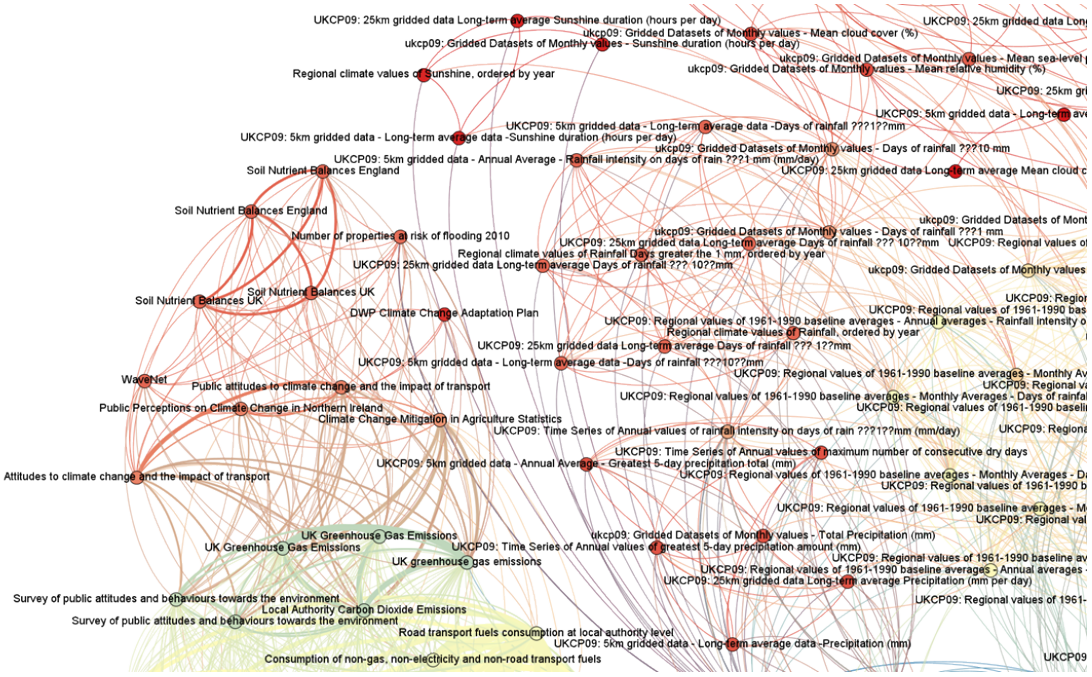
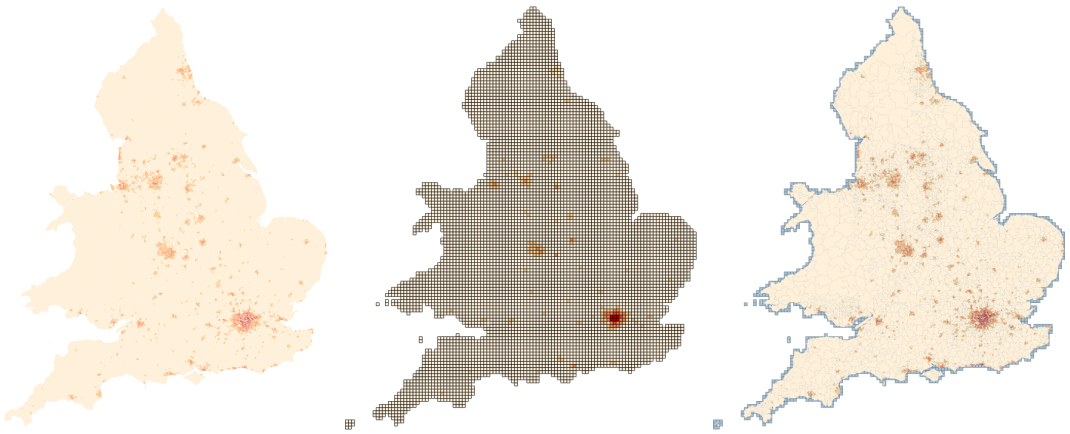


Figure 7.16: Keyword graph built from the data.gov.uk data store showing all datasets with the 'climate' keyword.

Once the 'Data Store Miner' library has been used to automatically add all 2,558 Census 2011 maps to the MapTube website, then comparison between the Census and other maps using different geographies is possible. The upload process is achieved via a direct database insertion and file copy from data prepared automatically from the Census bulk download. Then a web service is added which is an endpoint to download the spatial data for any dataset stored on MapTube. This allows any user to create an additional tool with the ability to pull the vector data behind any MapTube maps and compare them. By creating the template code and publishing it as open-source, other researchers can experiment with alternative geographic comparison schemes. Additionally, the website can now run a process which performs its own comparisons overnight, maintaining its own table of potentially related data. The

[11]The video of the presentation is available at the following link: *http://eprints.ncrm.ac.uk/3667/*.



(a) MSOA Population Density. (b) Gridded Population Density. (c) Both.

Figure 7.17: Population Density QS102EW0003 showing the original MSOA data (a), the 8KM gridded data (b) and comparison between MSOA boundaries and grid cells (c).

code for this 'MapTubeExplorer' project can be downloaded from: *https://github.com/maptube/MapTubeExplorer*.

The 'MapTubeExplorer' tool works by downloading and staging the requested files locally. Then the two maps are gridded according to a user-defined grid size, for example 8000 metres, before being correlated. Where a grid square crosses one or more MSOA boundaries, the data value assigned to the grid square is linearly interpolated according to the MSOA area intersecting the grid. In this case, the correlation is a standard spatial cross correlation, but the intention is to provide researchers with a tool which can be expanded. As the computation of the intersection area between an MSOA geometry and a grid cell is time consuming, taking over 2 minutes per map in this example where there are 7201 MSOA areas and 6,943 grid cells, an improvement was made to the basic algorithm. In order to compute the correlation between a single test dataset and the 2,558 Census datasets which all use the same MSOA geometry, the concept of a template grid was added. This uses the MSOA boundary file to compute a list of area keys intersecting each grid cell and stores the area key along with the fraction of cell coverage. Each template cell then contains a list of MSOA areas covering the cell and an associated weight derived from the fraction of intersecting area. The grid data for any new datasets is computed on a cell by cell basis as the scalar product of the new data values and the vector of $(areakey, weight)$ tuples for all MSOAs covering each template cell. Using the pre-computed template when a new dataset is downloaded from MapTube results in the gridding process now taking just 15 seconds where it was over 2 minutes previously.

The three maps in figures 7.17a,b and c show the result of the gridding process using the population density variable, *QS102EW0003*, from the 2011 Census. The 8KM grid size has been chosen as it results in 6,943 grid cells (114x137), which is comparable to the 7201 areas in the MSOA level data, although the grid cells will over-represent large MSOA areas and under-represent the denser areas like London. Data cells which do not cover any of the map, for example in the sea or rivers, are not counted, so $114 \times 137 = 15,618 > 6,943$. The grid density is configurable in the command line of the program, so it is possible to explore the effect of grid density. The gridding process itself is computationally intensive due to the geometry manipulation involved in determining the percentage of coverage of an MSOA area covering a cell rectangle. With the 8KM grid, this process takes 95 seconds when running on an Intel i7-3770K (3.5GHz)[12]. An interesting parallel exists here between the tiled map processes covered in the previous chapters, which produce 256 pixel image tiles for online maps, and the data gridding process which is used here. Both cut maps into tiles, but the process used here requires the additional step of calculating the covered area from the raw vector data, which, in the case of the MSOA data, is originally 251MB in size. Downloading the data from MapTube and staging it on the local computer prior to analysis requires this data as input, so the process is also limited by network bandwidth.

Finally, it is now possible to compare Census based maps using the MSOA geography with data which is not Census derived. While, in theory, any dataset could be compared, the results are only valid for a variable which is continuously varying over all of the spatial areas due to the mathematics behind the correlation formula. This prevents comparisons with data based on discrete classes, although a different correlation formula could be devised. This was the original reason for designing a tool which could be modified easily, specifically to encourage this type of experimentation. The "DatastoreMiner" and MapTube map creation process both use the same hashing technique when they data mine the data. This allows MapTube to differentiate between a real number data field that is continuously varying and a class label field containing a limited number of discrete classes. The answer is in the cumulative density function of the data being examined.

The map in figure 7.18 shows the results of the UK 2010 General Election using data obtained from the Electoral Commission.

[12]The hardware is an Intel i7-3770K 3.5GHz CPU with 1TB solid state drive, 16GB of RAM and running on Windows 10. The MapTubeExplorer tool is compiled with .net 4.5.2.

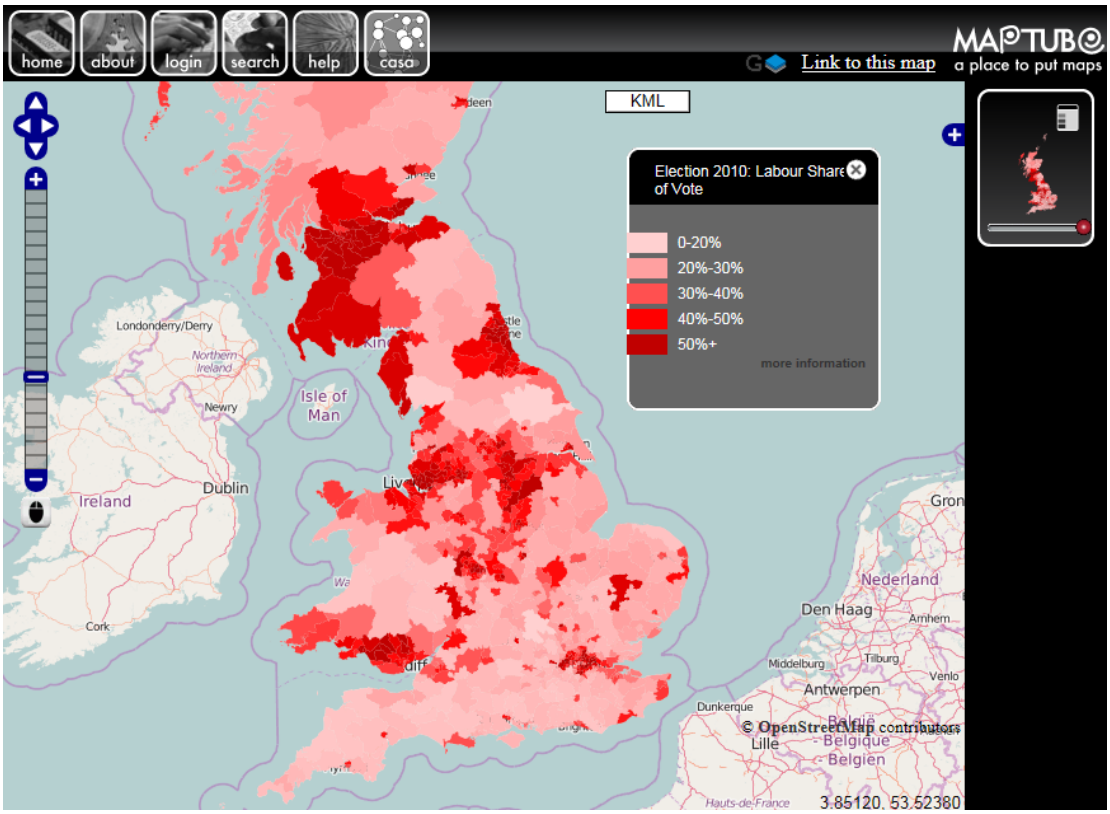


Figure 7.18: Labour percent of vote from the 2010 General Election. This map is available online at the following address: *http://www.maptube.org/map.aspx?mapid=840*.

The data shows Labour's percentage of the total vote based on the Westminster Constituencies geography. The Census data previously analysed uses a different geography, so this can be used to provide a first test of the tool. After a spatial cross correlation was performed between the Labour voting data and all 2,558 Census data variables, the results are shown in the graph of figure 7.19. The maximum correlation value of $I = 0.26$ reflects the fact that the election data contains Scotland, while the Census data does not.

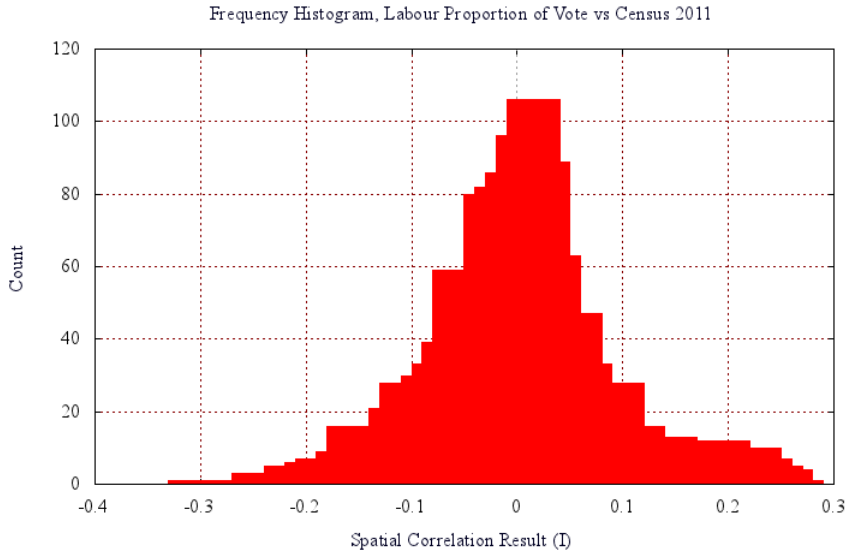


Figure 7.19: Frequency histogram showing Labour percent of vote from the 2010 General Election correlated against all the Census 2011 variables at MSOA level. The x axis shows the spatial correlation value (I), while the y axis is the frequency of occurrence. The total sample is 2558 variables.

Grid cells which are not present in both maps contribute zero to the overall correlation value, resulting in the low maximum value. The correlation is only performed on the areas of the two maps that overlap, but the scaling factor is calculated for the whole geographic area, resulting in the linear scaling down of the maximum correlation factor seen here. As the rank of the overall correlation is the factor being analysed, the scaling is not an issue. An alteration to the methodology could be made to account for this scaling factor automatically, but, ideally, the Scottish Census data could be added. Given the problems of the England and Wales and Scottish Census data being split, this would require significant work, so has not been attempted at this point. However, the results show some interesting trends. The top 11 positive correlation results are listed in table 7.6, with the full results available online at the following address: *http://www.maptube.org/PhDData/CH7-DataExplorationDataStoresAndCorrelation/labproportion-census-correlation.csv*.

Geographic knowledge discovery of this type using data from data stores is a potentially useful technique, allowing many questions to be asked of the data. For example, by limiting the analysis to only datasets containing data on ethnicity or country of origin, a profile of voters could be built. While space prevents a rigorous analysis of results from using the MapTubeExplorer tool, it forms an essential bridging component by providing analysis of data using different geographies. This is required in the next

I	mapid	Description
0.264	2398	Unemployed: Never worked (KS602EW0028) from the 2011 Census table Economic Activity - Males (KS602EW)
0.259	2384	Unemployed: Never worked (KS602EW0014) from the 2011 Census table Economic Activity - Males (KS602EW)
0.255	2369	Unemployed: Never worked (KS601EW0028) from the 2011 Census table Economic Activity (KS601EW)
0.254	2212	Day-to-day activities limited a lot: Age 16 to 64 (KS301EW0020) from the 2011 Census table Health and Provision of Unpaid Care (KS301EW)
0.250	4162	L13.2 Routine production occupations (QS609EW0046) from the 2011 Census table NS-Sec of Household Reference Person - People Aged under 65 (QS609EW)
0.250	2380	Economically inactive: Long-term sick or disabled (KS602EW0010) from the 2011 Census table Economic Activity - Males (KS602EW)
0.249	2355	Unemployed: Never worked (KS601EW0014) from the 2011 Census table Economic Activity (KS601EW)
0.249	2682	8. Never worked and long-term unemployed (KS612EW0026) from the 2011 Census table NS-SeC - Males (KS612EW)
0.247	2197	Day-to-day activities limited a lot: Age 16 to 64 (KS301EW0005) from the 2011 Census table Health and Provision of Unpaid Care (KS301EW)
0.246	2351	Economically inactive: Long-term sick or disabled (KS601EW0010) from the 2011 Census table Economic Activity (KS601EW)
0.246	3825	Economically inactive: Long-term sick or disabled (QS601EW0015) from the 2011 Census table Economic Activity (QS601EW)

Table 7.6: Top 11 positive correlations between Labour percentage of vote and the Census 2011 tables. The *mapid* column identifies the map on the MapTube website. The url *http://www.maptube.org/map.aspx?mapid=2398* references the original map data for the first line of this table. The full results are online at the following address: *http://www.maptube.org/PhDData/CH7-DataExplorationDataStoresAndCorrelation/labproportion-census-correlation.csv*

section which deals with real-time data and its relationship with the type of static data analysed here.

7.1.4 Conclusion

The initial aim behind this research is to build an information system capable of analysing a block of connected data rather than just individual datasets in isolation. In conjunction with the current proliferation of data stores and the quantity of open data now becoming available, the ability for a GIS system to be able to pull in data from the Internet and perform the pre-processing automatically is a high priority. In addition to this, deriving structure from large quantities of data is an aid to understanding and has the secondary benefit of making it easier to search through these large archives of information. While this section has only dealt with a few comparison and correlation techniques, the aim was always to look at a framework which could be extended. No single technique is ever enough, so geographically weighted regression (GWR), hierarchical agglomerative clustering and other spatial analysis and data mining techniques can be included to give a broader range of options. The important point to bear in mind,

though, is that these are outlier detection techniques designed to find where there *might* be a causal relationship. The data mining and correlation on its own can never prove the relationship, that needs to be done using information from the real world.

In terms of showing how one dataset fits in with all the others, the opposite point might actually be the more interesting one. There is no way of knowing whether an important piece of information is missing from the complete picture of urban datasets. By using structure and linking data together, there is a case to be made for further analysis of any datasets that are outliers. If a dataset about a city or country does not relate to anything else, then perhaps there is a missing link somewhere? This is similar to the idea of self-similarity, by defining something in terms of how it relates to everything else.

After analysing static data, the following section extends this to look at real time city data and how similar analysis techniques can be applied with the additional element of time.

Chapter 8

Data Exploration: Real-time and Data That Moves

This chapter expands on the previous work of Chapter 6, "Real-time Mapping and Agent Simulations", using the agent-based modelling framework developed for the London Underground and bus networks to analyse the real-time data collected about London. After analysing the rush-hour peaks in vehicle numbers, the chapter then looks at the self-similarities present in the data and how they can be used for calibration, before adding the U.K. rail network data. Looking at interactions between connected systems, environmental data in the form of weather and air quality are included, leading to an analysis of how these systems interact.

8.1 Real-time and Data that Moves

Drawing on the London Underground case study in the previous chapter, using real-time sources of data along with 3D visualisation enables the data to be analysed in a virtual representation of its original context. This, coupled with predictive models grounded in archive data and data mining techniques, build into a real-time GIS capable of analysing what is happening in a city right now. By using the predictive properties of modelling, it is also possible to experiment on a real city and ask "what if?" questions rooted in real data.

Using the software infrastructure, distributed computing and application service stack introduced in the previous chapters, is is now possible to analyse real-time city scale data. Figure 8.1 shows all the currently identified sources of data about London, the UK and the World.

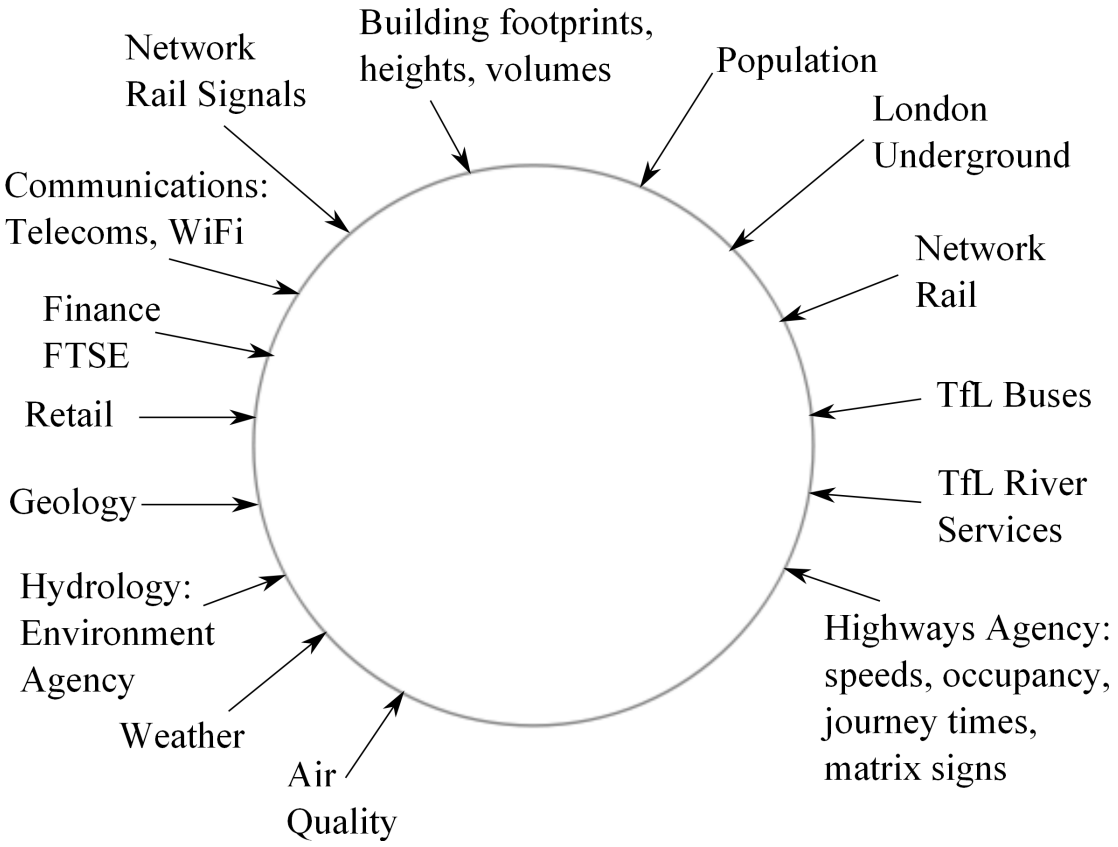


Figure 8.1: Real time data sources containing data about London.

The data sources shown in figure 8.1 can be grouped into categories based on frequency of data, geographic extents, tangible or intangible and proxy data. The 'London Underground', 'TfL Buses', 'River Boats' and 'Network Rail' data all describe positions of actual vehicles and so represent tangible objects in the real world. The 'Highways Agency' traffic counts are a proxy for real vehicles and so represent intangible sensor data. The temporal resolution of all these datasets is higher than any of the other sources of data. For the sensor data, the weather and air quality sensor networks are geographically more sparse and deliver data at an hourly resolution[1]. Hydrological data also falls into this group, with river levels obtainable from the Environment Agency in real time. 'Communications', 'Retail' and 'Finance' have been added as sources of data which are required but not yet obtainable.

Visualisation is possible at either the macro scale or micro scale, and at varying degrees of temporal resolution. Time and space are inextricably linked in any dynamic visualisation because time determines how fast objects move through the dimensions of the space. This gives rise to perceived speed as seen by the viewer. As an example, if a

[1]Crowd sourced weather data is also available which can give a much higher temporal resolution, but at the expense of an unknown accuracy.

map of London occupies 1920x1080 pixels on the screen and the London Underground tubes are displayed at their true speed ($\approx 9ms^{-1}$, or 20MPH), then each object will move at 0.79 pixels per second in the animation. When zoomed in to street level, where the screen covers just 200 metres, this then equates to 86.4 pixels per second[2].

The inclusion of sparse resolution temporal data such as weather and air quality requires intelligent interpolation, as does sparse resolution spatial data, which is where modelling meets geographic information systems. With the data available, each data source can be represented using the latest data and modelled using predictive systems which are shown to work with the archived data. Each data source represents a system which interacts with other systems.

While the London Underground network has already been covered in detail, the GeoGL software is now applied to the visualisation and analysis of the remaining transport networks. Real-time data for London's buses and Network Rail trains for the whole UK are readily available. Both of these sources can be used to identify the locations of physical vehicles, while road traffic data is split between the TfL delay messages covering London and the Highways Agency sensors covering the rest of the country. Neither can be used to identify physical road vehicles, with the Highways Agency data giving traffic flow from counts and average speed sensors, while the TfL data only contains delay messages making the two forms of data incompatible with each other and with the other transport data.

Using the archive of 2014 tube and bus data for London, it is now possible to analyse whether there is any variation in commuting times across the two modes of transport. Both the tube and bus data sources only contain vehicle numbers over the day, with passenger data requiring access to TfL Oyster card data showing 'tap ins' and 'tap outs' of actual commuters as they enter and exit the stations. As this is only available for limited days during the year, a comparison of the vehicle numbers between bus and tube is made first, with the results then analysed in relation to passenger station entry and exit data for 2014. Network Rail data is included later due to the problems of gaps in the data and having to define a London area for data covering the whole of the UK.

The data contains 365 days at 3 minute resolution for tube and bus, resulting in $365 \times \frac{24 \times 60}{3} = 175,200$ time points. Each day belongs to a class of $\{Saturday, Sunday, Weekday\}$ which fits the timetables in place for the relevant parts of the week and

[2]Calculation based on the width of London begin 35.8 miles and tube speeds measured by averaging the real-time data.

allows for comparison of the data. By using the 3D visualisation and the tube and bus models used for real-time animation, a correlation model can be built that loads each successive animation frame, performs a correlation between the tube and bus numbers, writes the results to a log file and then continues with the next day. Equation 8.1 shows the cross correlation function used to find the time shift.

$$R_{xy}(\tau) = R_{xy}(mT) \approx \frac{1}{N} \sum_{1}^{N} x(n)y(n+m) \tag{8.1}$$

(T=sample period, N number of samples, [Lyn91, pp116])

In equation 8.1, X is the tube and Y is the bus, so any time shift, m, between the two is the bus times relative to the tube times.

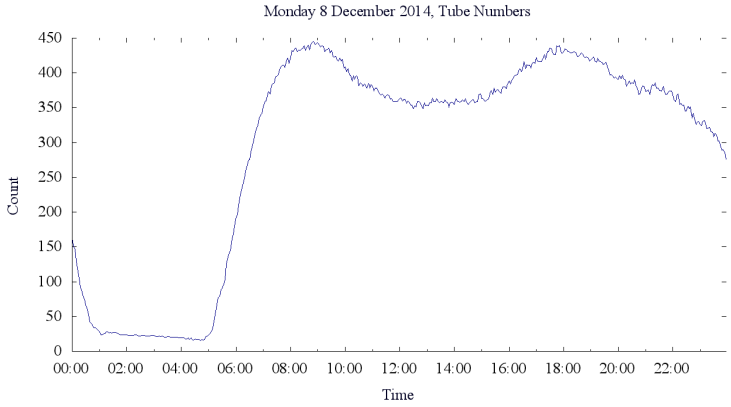
Taking a single day as an example, the data for Monday 8th December 2014 is shown in figures 8.2a (tubes) and 8.2b (buses). The resulting correlation between the two is shown in figure 8.2d. The maximum value of the correlation function is taken as the minimum error between the two modes of transport with an associated time shift, mT. Also included is the ratio of buses to tubes running throughout the day, figure 8.2c, as this highlights when the timetable patterns change and the numbers go up and down to satisfy demand.

With the simulation set up to load vehicle data from files in near-line storage created by processing the raw data using the ANTS library, the 3D visualisation is used as a final verification before being switched off completely to allow the non-visual part of the processing to run at maximum speed. The computation is limited by disk based I/O rather than algorithmic intensity, due to the amount of data loaded from near-line storage. Table 8.1 shows the statistics related to the amount of data in the archive used to generate the results for the year.

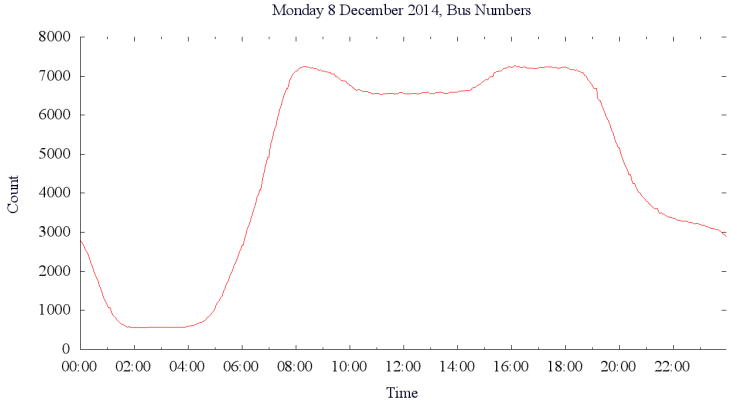
Method	Data Size	Time Points	Missing Data	% Missing
Tube	92.2GB	173,517	1,683	0.96%
Bus	404.3GB	160,794	14,406	8.22%

Table 8.1: Tube and Bus Number Statistics for 2014. There are 175,200 data points in a full year.

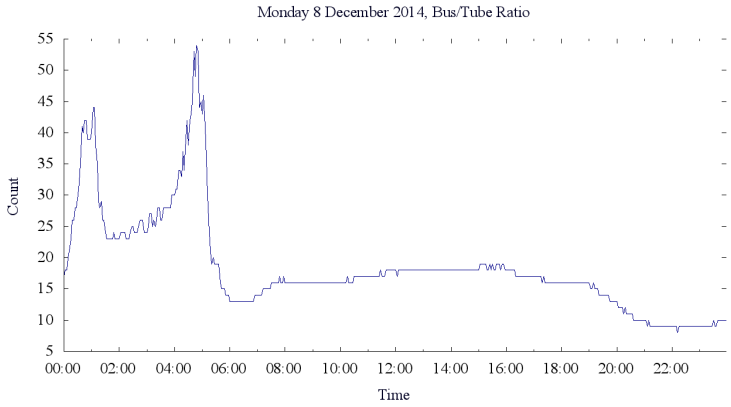
While 0.96% of data missing for the tube over a year looks significant, it misses the fact that the tube data is an aggregate of ten tube lines, all of which need to be missing for a time point to be skipped completely. With the bus data, the 8.22% loss reflects the greater complexity, with a permanently connected stream required for processing



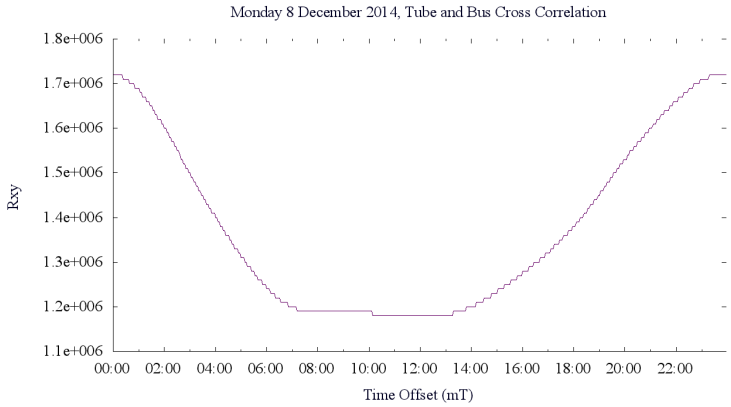
(a) Number of tube trains.



(b) Number of buses.



(c) Bus : Tube ratio.

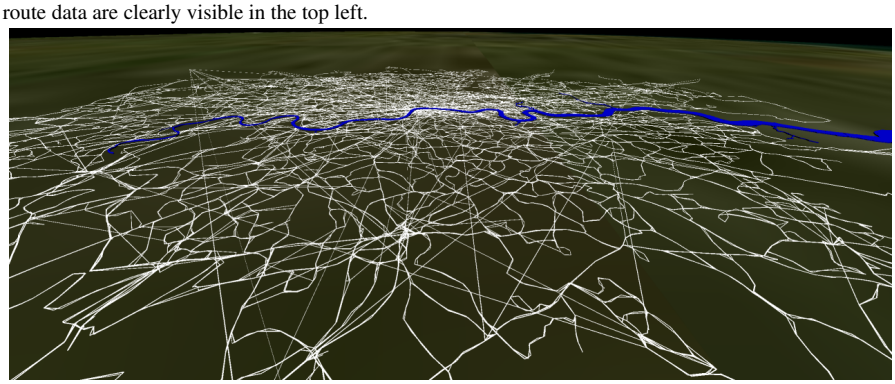


(d) Correlation factor between (a) and (b), Rxy.

Figure 8.2: Monday 8th December 2014, number of tubes, number of buses, the ratio of buses:tubes and correlation coefficient, Rxy, throughout the day.

bus movement messages in real-time. In addition to this, the data transfer is also more 'noisy', with frequent errors in the 'JSON' format stream causing problems with the message parsing. Figure 8.3 shows the complexity of the bus network.

(a) Bus routes seen looking directly down on London. The errors in the network from the official TfL route data are clearly visible in the top left.



(b) Bus routes looking North towards London at a low angle to the horizon. The curve of the Earth is visible in this shot.

Figure 8.3: Two pictures of a complex transport network. There are 21,987 bus stops (agent nodes), 53,896 route points (links), approximately 7,000 live buses (moving agents) and one river. Currently, 409 bus stops can be identified as missing from TfL's master table, demonstrating one of the problems with data at this scale and why building origin/destination links from the real-time data is important.

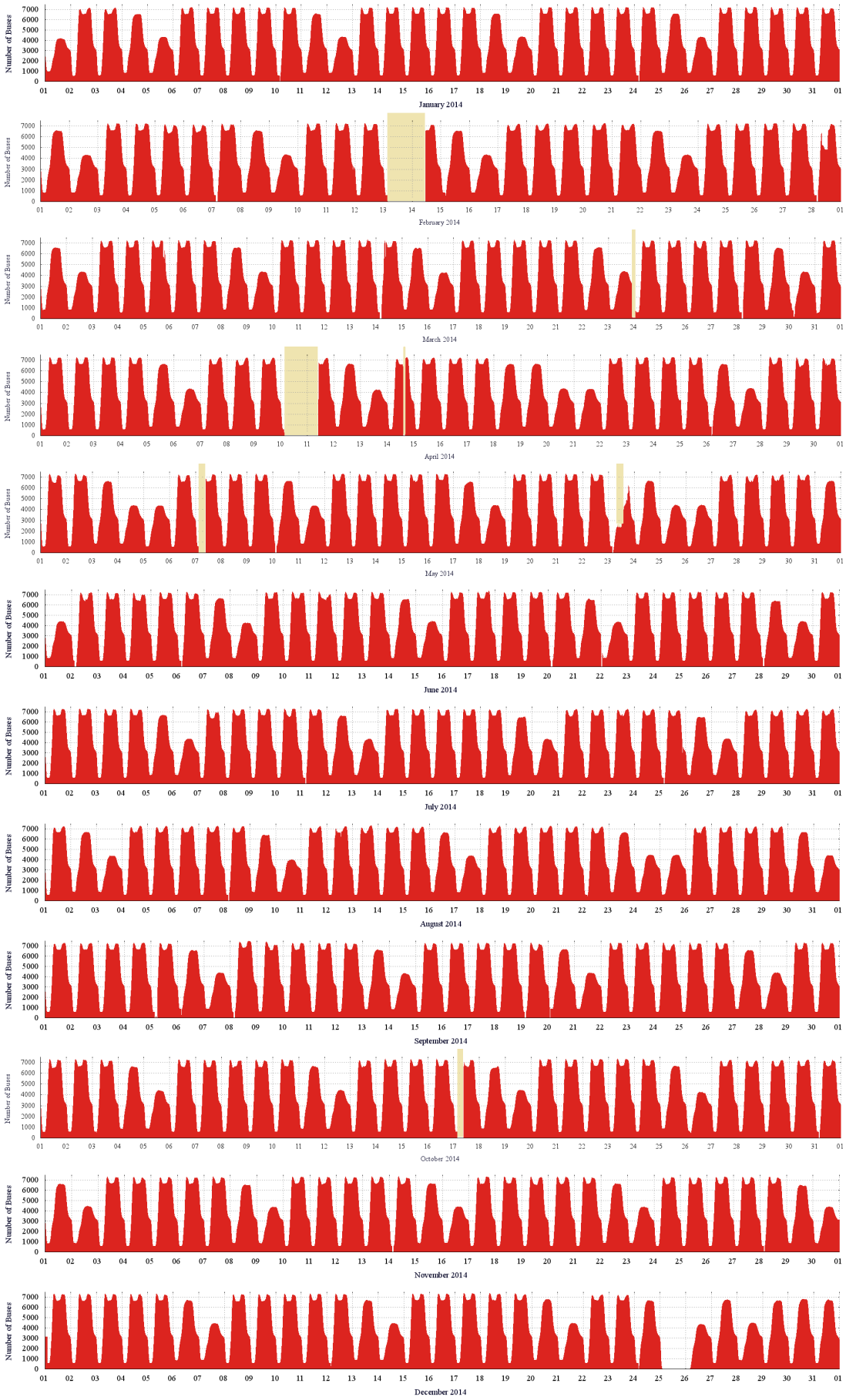


Figure 8.4: Bus Numbers for 2014. Yellow highlighting indicates data loss.

Using all the bus data collected for 2014, figure 8.4 plots bus numbers for the entire year. Similar to the tube data shown earlier in figure 6.6 of chapter 6, this gives a visual indication of the morning and evening rush hour peaks, plus any irregularities in the service.

Moving on to the analysis of all the data for 2014, figure 8.6 shows the plot of the correlation data for all 365 days. The compute time for the whole year of data was 2 hours, 5 minutes and 45 seconds, or 7540 seconds in total.

The raw data and frequency distribution shown in figures 8.5 and 8.6 suggest a definite lag averaging 11.5 minutes with the bus numbers lagging behind the tube. The standard deviation of $\sigma = 36.33$ is high, being influenced by the outliers in the data. As neither the Saturday nor the Sunday data shows the characteristic morning and evening peaks in commuter services, the next approach is to calculate the statistics based on just weekday data. This results in a mean of $\mu = -14.25$ and standard deviation of $\sigma = 41.18$, while retaining all the outlier data for a realistic comparison with the previous data. As there are only 365 data points, the outliers can have a significant effect on the final results, so they need further investigation as specific failure cases. The data presented here shows a definite lag with the tube numbers following the bus numbers by an average of 14.25 minutes over the year.

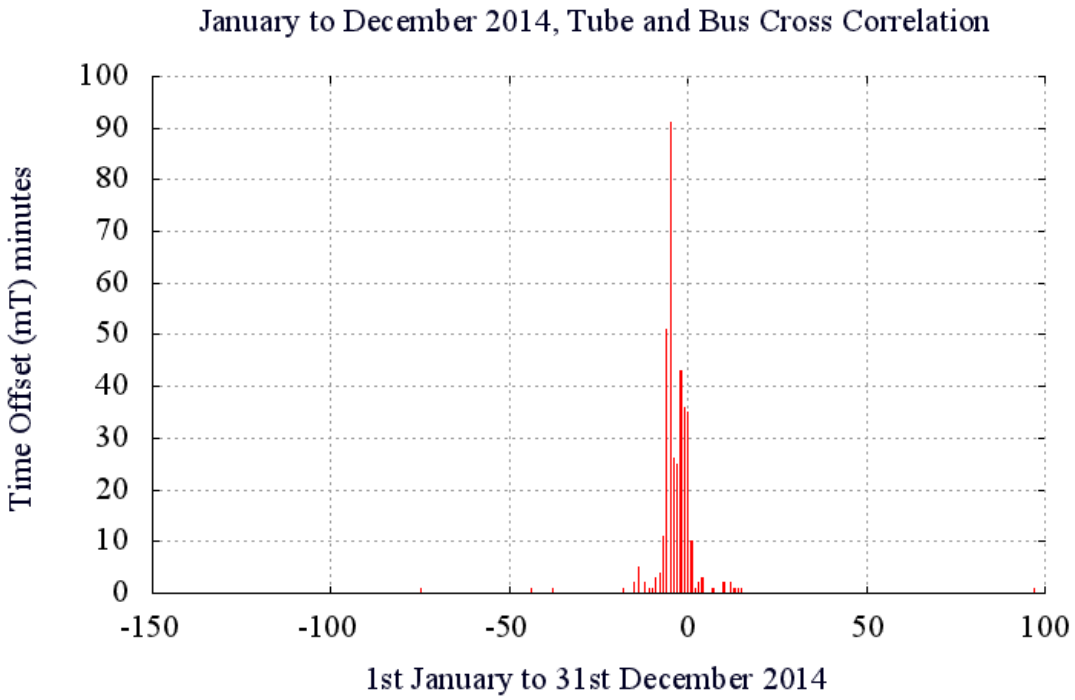


Figure 8.5: 1st January to 31st December 2014, correlation frequency histogram between tube and bus numbers. Weekend and weekday data is included together.

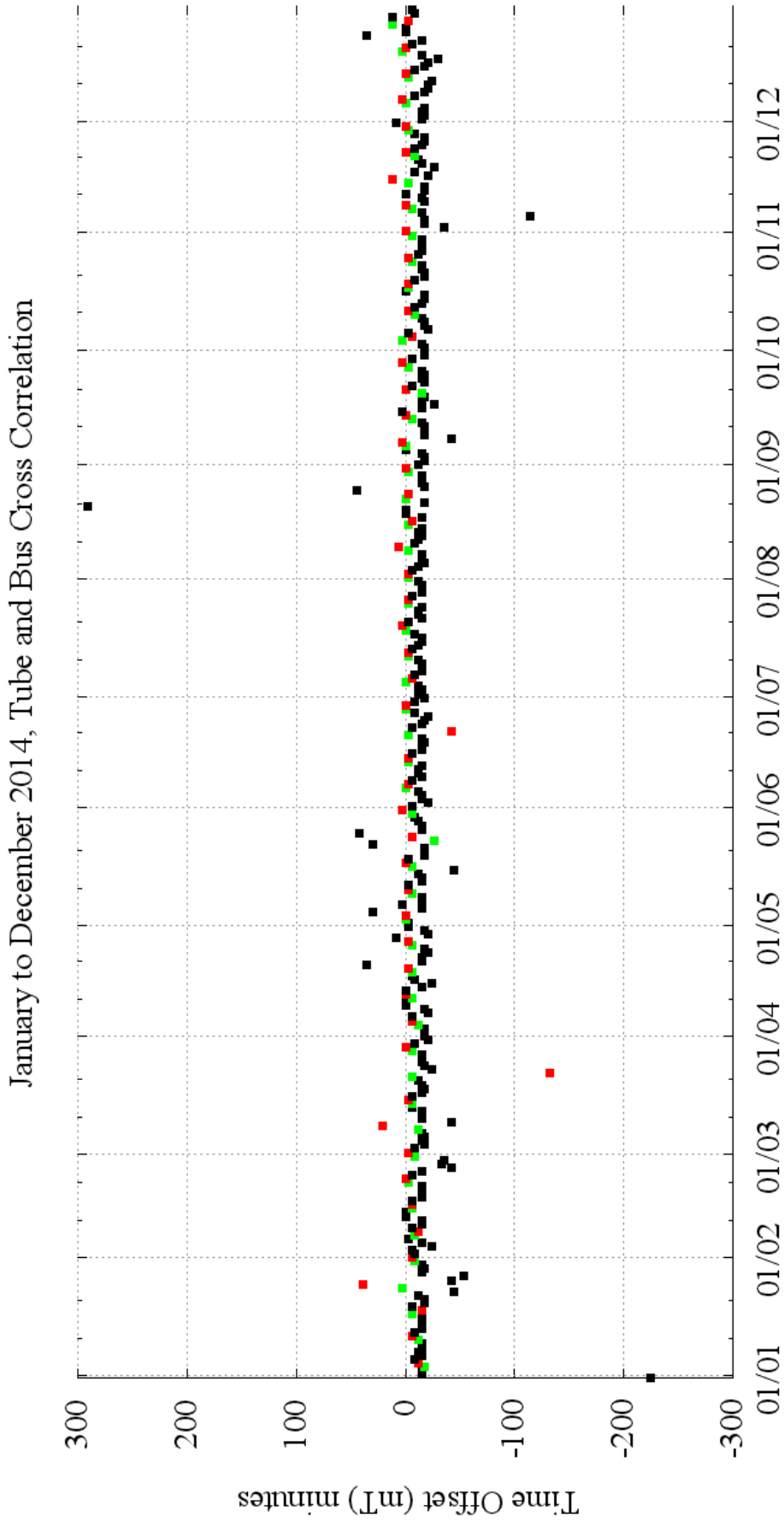


Figure 8.6: 1st January to 31st December 2014, correlation between tube and bus numbers. This plot shows all days of the week, $\mu = -11.5$, $\sigma = 36.33$. Red=Sunday, Green=Saturday, Black=weekday.

Looking at the data more closely though, the characteristics of the morning and evening peak rush hour data appear to be different. The morning peak occurs around 8am and is over by 10am, while the evening peak begins around 4pm and only finishes at 7pm. This is apparent in the graphs of bus and tube numbers plotted in figures 8.2a and 8.2b. By employing a windowing function and filtering the data into morning and evening groups, the correlation technique can be repeated to check for any differences. The use of window functions is a standard technique is signal analysis, where a section of a time domain signal is transformed into the frequency domain using a Fourier transform in order that the major frequency components comprising the signal can be isolated. Due to the components of the Fourier transform implicitly assuming that the partial section of the signal repeats from minus infinity to plus infinity, where the beginning and end of the sample are stitched, any discontinuities give rise to high frequency noise. In the example here, the bus and tube numbers are the time domain signal, which exhibits a characteristic morning and evening rush-hour peak. However, the features in these two peaks differ markedly, and an attempt to perform a time domain cross correlation results in an average somewhere between the two. The way to get around this problem is to analyse the morning and evening data separately, which requires untangling the two signals from the time domain data. This is where the windowing function comes in, allowing the attenuation of the PM data from the AM and vice-versa. The windowing function used is a Hamming window [Lyn91, pp199] with parameters chosen to separate the data into the required morning and evening groups. Equation 8.2 is the weighting function applied to the bus and tube data, which is plotted in figures 8.7 and 8.8. The weights shown by the red and blue lines are used to separate the bus and tube numbers into AM and PM samples, which are then handled individually.

$$w(n) = 0.54 + 0.46 \cos \frac{(n - C)\pi}{N} \tag{8.2}$$

In this instance, $C = 160$ and $N = 240$ for the morning filter, while $C = 340$ and $N = 300$ for the evening filter. The offset, C, is chosen to centre the filter on 8am and 5pm respectively, while the width, N, is chosen to ensure that the filter cut-off is wide enough to cover the whole of the peak period. The variable n is the sample number.

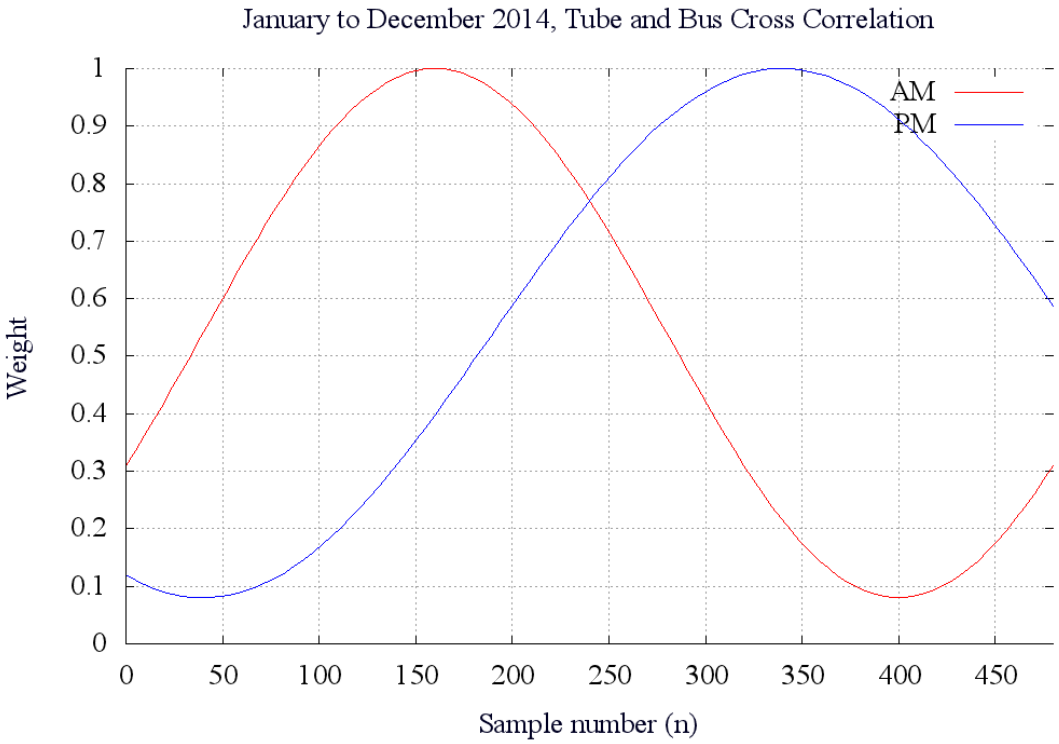
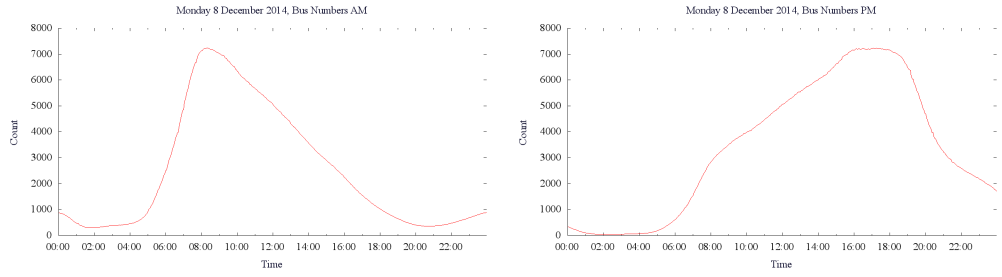
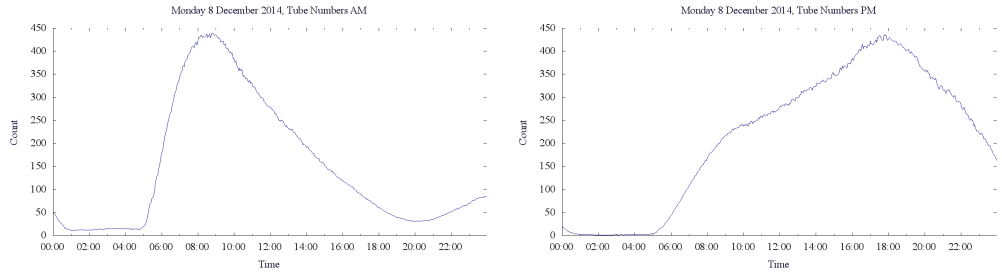


Figure 8.7: Filter window used to weight bus and tube numbers for AM (red) and PM (blue) analysis.



(a) Bus Numbers Weighted AM.

(b) Bus Numbers Weighted PM.



(c) Tube Numbers Weighted AM.

(d) Tube Numbers Weighted PM.

Figure 8.8: Bus numbers (top) and tube numbers (bottom) for morning and evening rush hours using data from Monday 8th December 2014.

Now, when the simulation is run with the new correlation metrics, the morning and evening data in figures 8.9 and 8.10 show different results.

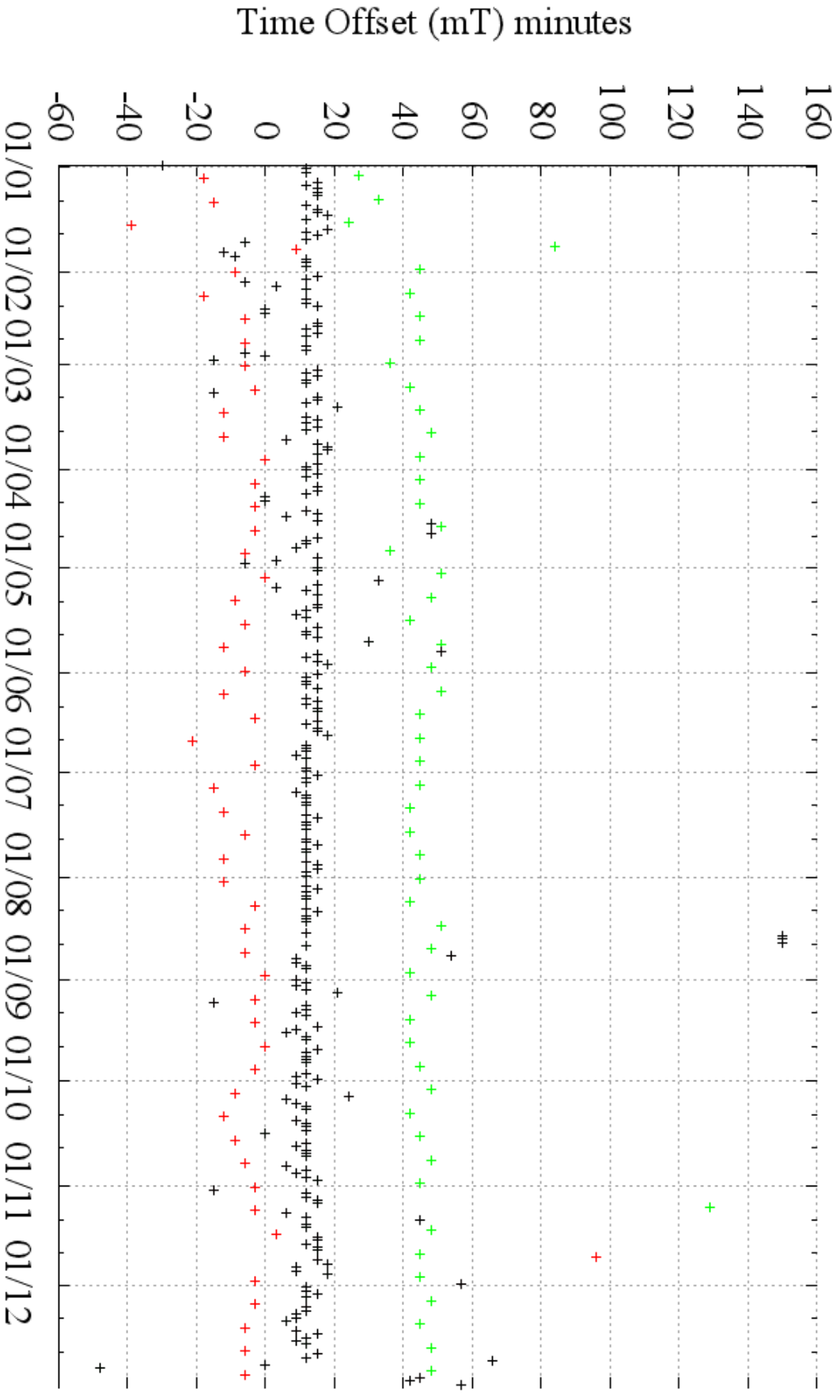


Figure 8.9: Bus to tube correlation of vehicle numbers partitioned into the morning rush hour peak, Sunday (RED) $\mu = -5.19$, $\sigma = 15.83$, weekday (BLACK) $\mu = 14.228$ $\sigma = 18.16$, Saturday (GREEN) $\mu = 46.67$ $\sigma = 13.78$.

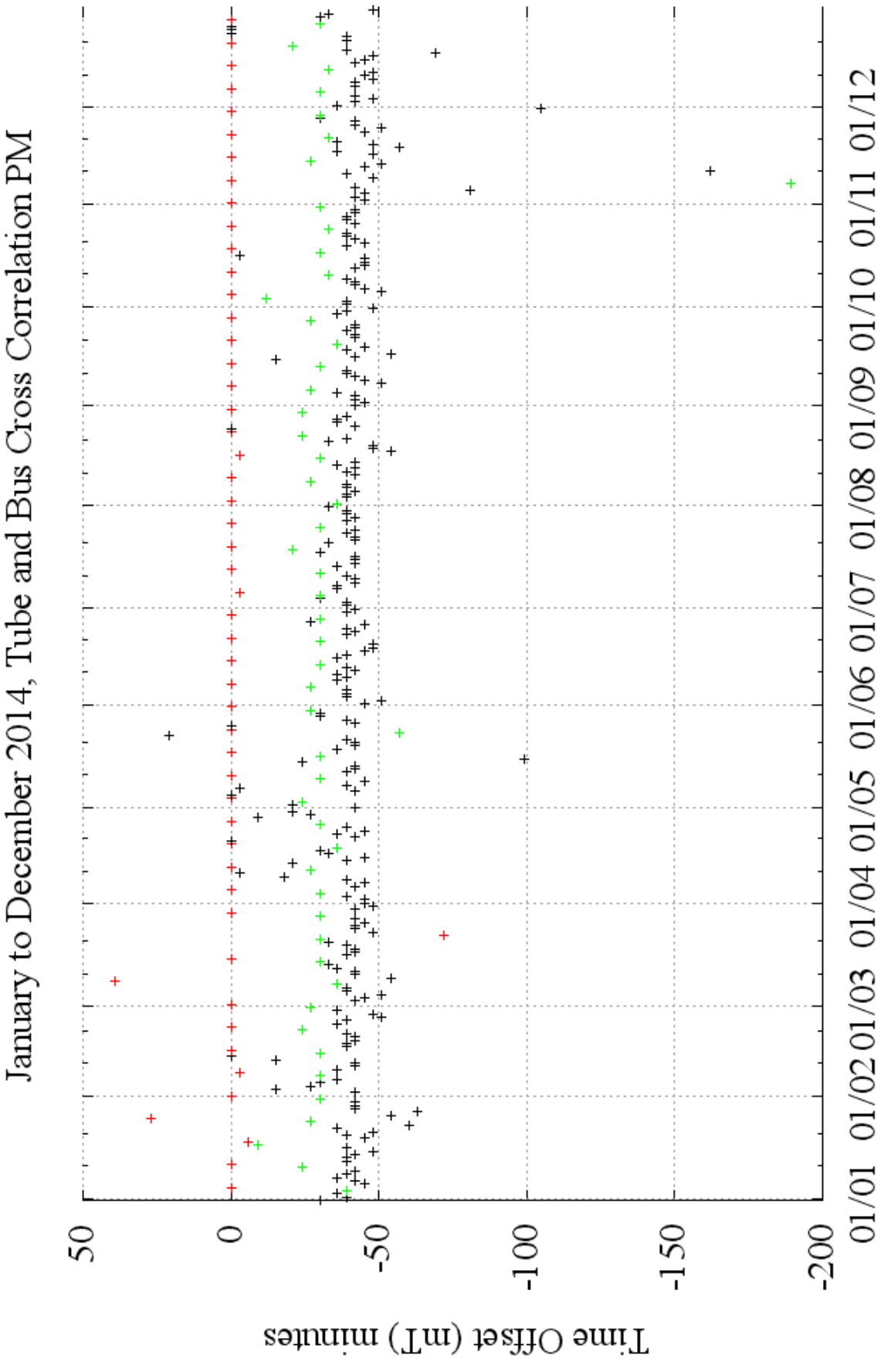
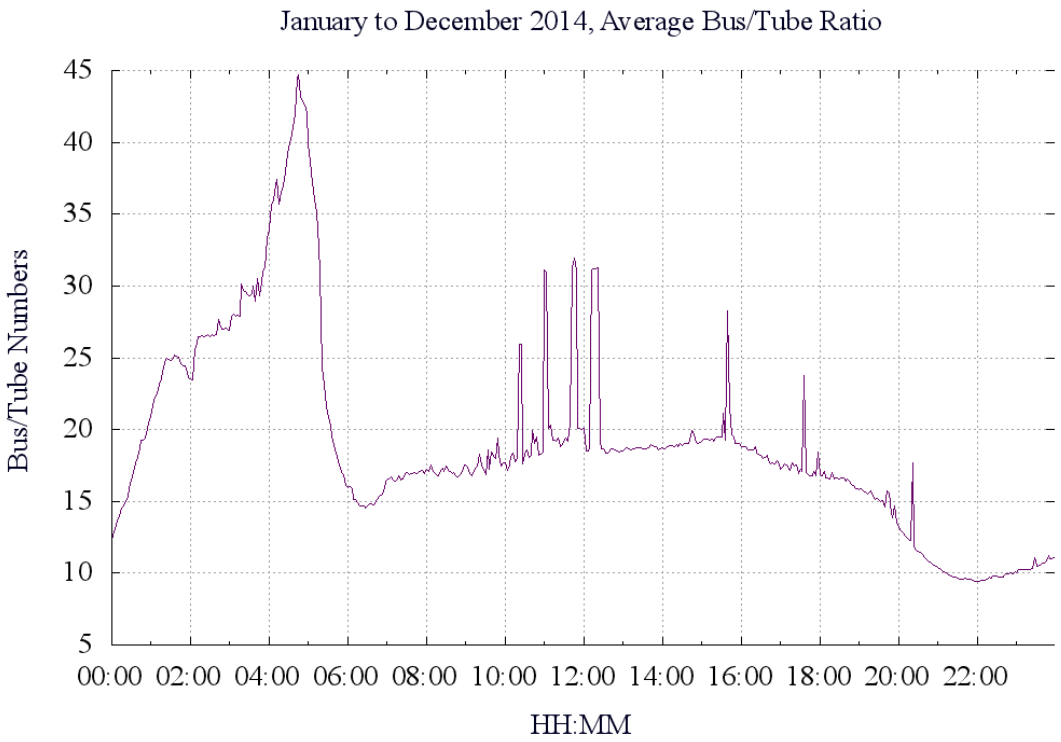


Figure 8.10: Bus to tube correlation of vehicle numbers partitioned into the evening rush hour peak, Sunday (RED) $\mu = -0.40$, $\sigma = 12.0$, weekday (BLACK) $\mu = -39.62$ $\sigma = 14.84$, Saturday (GREEN) $\mu = -32.25$ $\sigma = 22.87$.

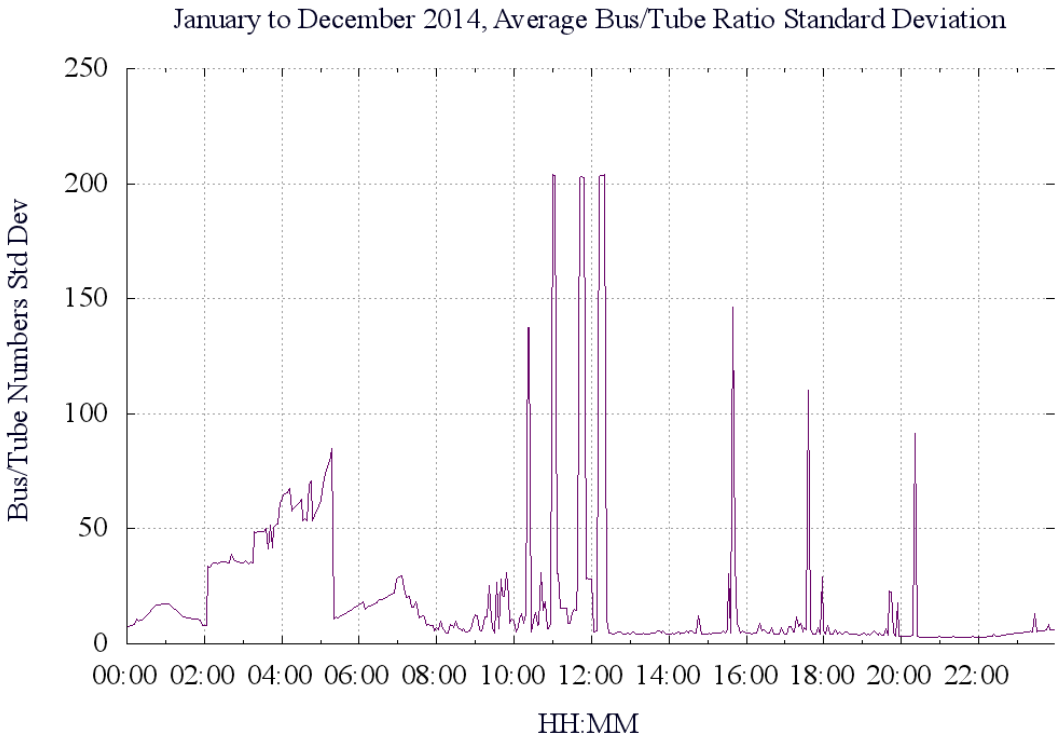
The points on the scatter plots have been coloured to indicate Saturday, Sunday or weekday data, which immediately shows how the data for Sunday does not exhibit any significant time lag, with a mean of -5 minutes for the morning data and -0.4 minutes for the evening data. There are no discernible rush hour peaks in either the bus or tube for Sunday, so correlation here is dependent on the leading edge or falling edge transition to the steady state. For the weekday data, though, a pattern emerges with a mean of 14 minutes in the morning and -39 minutes in the evening, suggesting the tube peak time begins before the bus in the morning, but, in the evening, the situation is reversed with the bus numbers ahead.

Finally, for this section, the average Bus:Tube ratio is plotted as an aggregate for the whole year in figure 8.11a with the associated standard deviation in figure 8.11b. The most prominent feature in this data is the 4am to 5am peak when the tube network shuts down and the ratio to the number of buses increases. Also of interest are the four peaks between 10am and 12pm and the three peaks from 3pm until just after 8pm. These also correspond exactly to high periods of variability in the data as shown in figure 8.11b. As these are averages over a whole year of data, the suggestion is that these points correspond to timetabling artefacts when either the bus or tube services are changing in frequency.

The $\frac{bus}{tube}$ ratio from figure 8.11a partially shows this trend towards greater bus capacity in the evening and a stretched evening commute, but is harder to interpret than the earlier correlation scatter plots. One final thing to mention relates to the proposal by London Underground to run late night tube services on selected parts of the network from August 2016. This is expected to alter the bus to tube ratios for the evening and alter the commuting pattern.



(a) Bus/Tube Ratio as an average for 2014.



(b) Bus/Tube Ratio Standard Deviation for 2014.

Figure 8.11: Bus/tube ratio calculated as an average of the time series for every day in 2014. Figure 8.11a shows the mean ratio across a day, while figure 8.11b shows the standard deviation.

8.2 Similarity, Feature Detection and Stream Mining

The data presented so far on the bus and tube networks show a signature morning and evening peak imposed by the commuters using these systems to get to and from work. This signal is also present in the individual tube lines and bus routes themselves, which can act as a feature indicating where and when problems are occurring. The reason for building a real-time transport model was to create a tool which allows the exploration of these features in the data more easily, as it is not a function of a regular GIS to extract data from temporal streams. The tube and bus models which parse the real-time streams in this application have been designed to perform a 'stream mining' function [WEH11, Ch9.3]. Formula 8.3 and 8.4 shows an implementation of the recurrence formula ([Knu00a, pp232], [Wel62] and [CGL79]) which is used to compute means and variances on expected wait times for every tube station, line, platform and bus stop. Data is produced on a daily basis, with this then aggregated to months.

$$M_1 = x_1,\ M_k = M_{k-1} + \frac{(x_k - M_{k-1})}{k} \quad \text{(M=mean)} \tag{8.3}$$

$$S_1 = 0,\ S_k = S_{k-1} + (x_k - M_{k-1})(x_k - M_k) \quad \text{(S=sum of squares)} \tag{8.4}$$

$$\sigma = \sqrt{S_n/(n-1)} \tag{8.5}$$

$$\mu_{a+b} = \frac{\mu_a n_a + \mu_b n_b}{n_a + n_b} \tag{8.6}$$

$$\sigma^2_{a+b} = \frac{n_a(\sigma_a^2 + (\mu_a + \mu_{a+b})^2)}{n_a + n_b} + \frac{n_b(\sigma_b^2 + (\mu_b + \mu_{a+b})^2)}{n_a + n_b} \tag{8.7}$$

By using expected waiting times based on the analysis of archive data, multi-modal visualisations of London's transport system are possible. Features like 'ratio of north-bound tubes to southbound tubes', 'probability of finding a tube at a specific station', 'number of tubes at station' and 'average distance between tubes'. Are all features which can be extracted from the software and used for analysis. One feature which works particularly well is the probability of finding a tube at a station. For stations which are a terminus, this probability is very high. The reason for this is due to the

fact that a terminus usually has more than one platform and services wait for a few minutes before turning around and heading off again. This means that a terminus with no parked tube is a significant event and likely to be indicative of a gap in the service.

One of the problems in analysing this type of real time data is obtaining a baseline set of data to compare against. In one sense, there is no normal day on London's transport system, so to obtain an operating baseline involves analysing a large set of data. Stream mining is a data mining technique designed to consume a large amount of data without using increasing computing resources, which is how the daily mean and variance for expected wait times at stations outlined earlier were calculated. TfL's own service status data can be used to train a system to recognise problems by using supervised learning, or, based on the assumption that a failure is an unusual event, a system can be designed that is self-calibrating. By making the observation that all tube lines, train operating companies and bus routes have the same daily variation in their numbers of vehicles running over the day, it is possible to build a system that is self-calibrating in real-time.

The data in figure 8.12 is taken from the "Explorer" view of the WEKA data mining tool kit [WEH11]. This shows the correlation of the number of running tubes for every combination of two tube lines over the whole of the data for May 2013. The Waterloo and City line has been removed as there are only 5 trains on the line, which is not enough to show a strong correlation. For the rest of the lines, a strong correlation is suggested, which can be investigated further using linear regression.

Using the 10 fold cross validation linear regression tool kit in WEKA, the following equations are obtained:

$$
\begin{aligned}
V &= 0.6206J - 0.0237 \text{ (correlation coefficient=0.98)} \\
V &= 0.5243D + 0.4557 \text{ (0.91)} \\
V &= 0.5576C + 0.4669 \text{ (0.93)} \\
V &= 1.1563H - 0.7925 \text{ (0.901)} \\
V &= 0.8931M + 1.8094 \text{ (0.83)} \\
V &= 2.2579B - 3.2299 \text{ (0.91)} \\
V &= 0.363N + 3.2262 \text{ (0.89)} \\
V &= 0.5927P - 0.9802 \text{ (0.951)} \\
V &= 4.7551W - 12.6042 \text{ (0.63)}
\end{aligned}
$$

The letters represent the number of tubes running on each line as follows: V=Victoria, J=Jubilee, D=District, C=Central, H=Hammersmith and City and Circle, M=Metropolitan, B=Bakerloo, N=Northern, P=Piccadilly, W=Waterloo and City.

The Victoria line (V) was used as the basis for all the equations, but the choice is arbitrary, as all 45 equations could be calculated if required. The aim here is to test the derived data against the rolling stock for each line, as specified in the TfL Key Facts documents[3].

Line Code	Line Name	Fleet Size
B	Bakerloo	32
C	Central	85
(H)	Circle	18
D	District	76 (68 District+8 Edgware Road)
H	Hammersmith and City	15
J	Jubilee	47 (Fleet has 59)
M	Metropolitan	49
N	Northern	84
P	Piccadilly	79
V	Victoria	37 (Fleet has 59)
W	Waterloo and City	4 (Fleet has 5)

[3]Transport for London. (2013, June 6). TfL Line Facts. Retrieved June 6, 2013, from Transport for London: *http://www.tfl.gov.uk/corporate/modesoftransport/londonunderground/keyfacts/1610.aspx*.

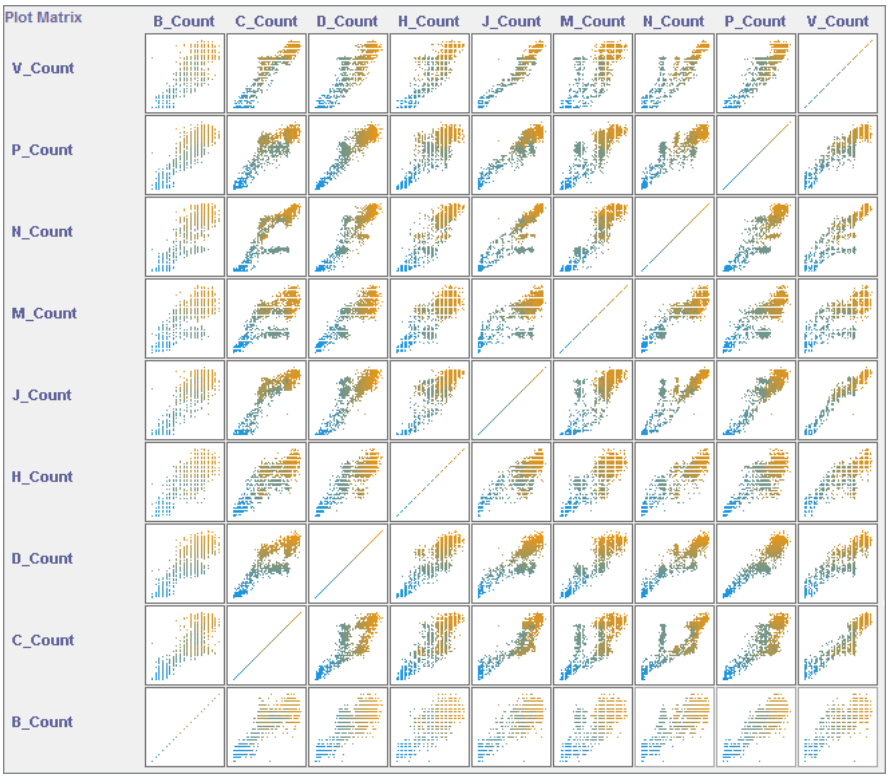


Figure 8.12: Tube numbers on all the London Underground lines used as a means of self-calibration. The ratios of tube numbers between lines follows the same daily variation and is related to the fleet numbers.

8.3 National Rail and Commuters

While the analysis up to this point has covered vehicle numbers and the supply side of the transport system, the missing link is currently the demand placed on the system by the commuters. At the present time, no real-time information about numbers of commuters is available, so the analysis can only take into account supply, with demand an unknown quantity. Although some Oyster card information is available, it is limited to only a few selected periods of 2012 and 2014. In order to make a full analysis, long periods of data need to be available, ideally covering the whole of the year without any gaps. The Census travel to work data is one possible alternative, as is the TfL data on station entry and exit numbers which was released under a Freedom of Information request. This contains data for every day and station in 2014, aggregated to 15 minute periods, which limits the correlation of commuter numbers with vehicle numbers. Figure 8.13 shows the monthly entry and exit averages for 2014, highlighting a clear morning and evening peak rush hour which is present in all the transport data. Even with these sources of data, predicting the effect of transport system disruption on commuters is very limited.

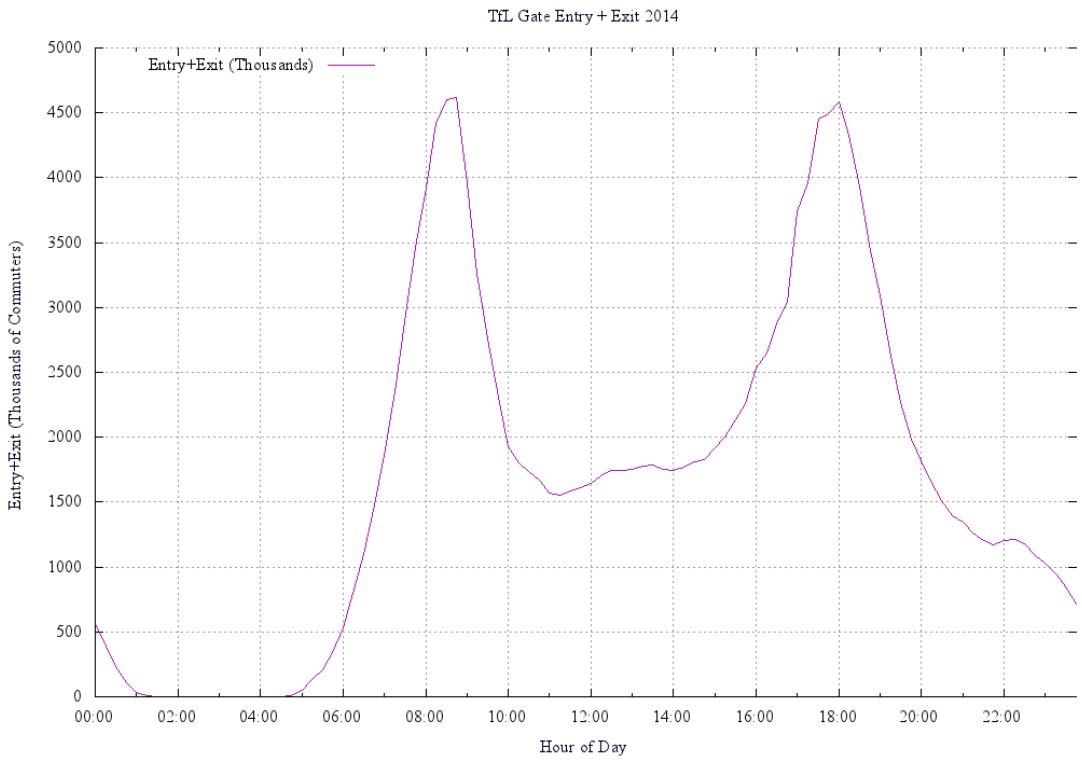


Figure 8.13: Tube entry and exit figures aggregated for all 2014 weekdays. The y axis shows the monthly average in thousands of commuters for all stations on the network by adding the entry and exit totals together.

In addition to this, data also exists for Network Rail trains, weather and air quality. The reason for not including the Network Rail data on overground trains with the tube and bus data is due to the problems in handling the data and differences in the information that it contains. There are also issues with determining the location of trains due to the fact that the timing points used for positions are not fully geo-located, which makes positioning trains difficult. As both the bus and tube data is limited to the London area, accurate positions of vehicles is unnecessary when counting numbers, but, when Network Rail data covering the whole of the UK is analysed, some method for determining which trains are in London is required for the counts to be comparable. One further problem is the fact that the data consists of real-time information from different train operating companies. The data for the 'Eurostar' company has been filtered out as it takes over 2 hours to get from London St. Pancras to Paris Gare du Nord, which is their only operating route.

There are additional problems with the Network Rail data, which complicate any analysis of train positions. The location in the train position message is based on a 'STANOX' code. There is currently no data on the actual positions of these timing points, which correspond to signals and stations where the trains are required to stop. In the case of a station, there is a link between 'STANOX' code and 'CRS' code, which enables the latitude and longitude to be found from the database of station locations. One solution to this problem is to use data-mining techniques to build a network of connections between the timing points and use the runlink times to interpolate between stations with known locations, but this has not been tried so far. As an indication of the problem, out of the total of $64,144,959$ trains counted in the 2014 data, $11,647,486$ contained a valid position, leaving $52,497,473$, or 81.84% with no usable location information. In the timing point location database there are 9113 unique 'STANOX' codes, with 2973 having a linked 'CRS' giving a position (32.6%).

Despite these problems, it is still possible to plot the daily variation for the year, taking into account some periods of high data loss as this source of data was not considered to be operational like the tube and bus feeds. The only additional requirement is to filter out any trains outside a box enclosing London. For this purpose, the London box is defined as follows: $((-0.51, 51.286), (0.34, 51.692))$.

Looking at the plot of train numbers throughout 2014 in figure 8.14, the periods of data loss are highlighted. Calculations from the raw data give $52,704$ data points for

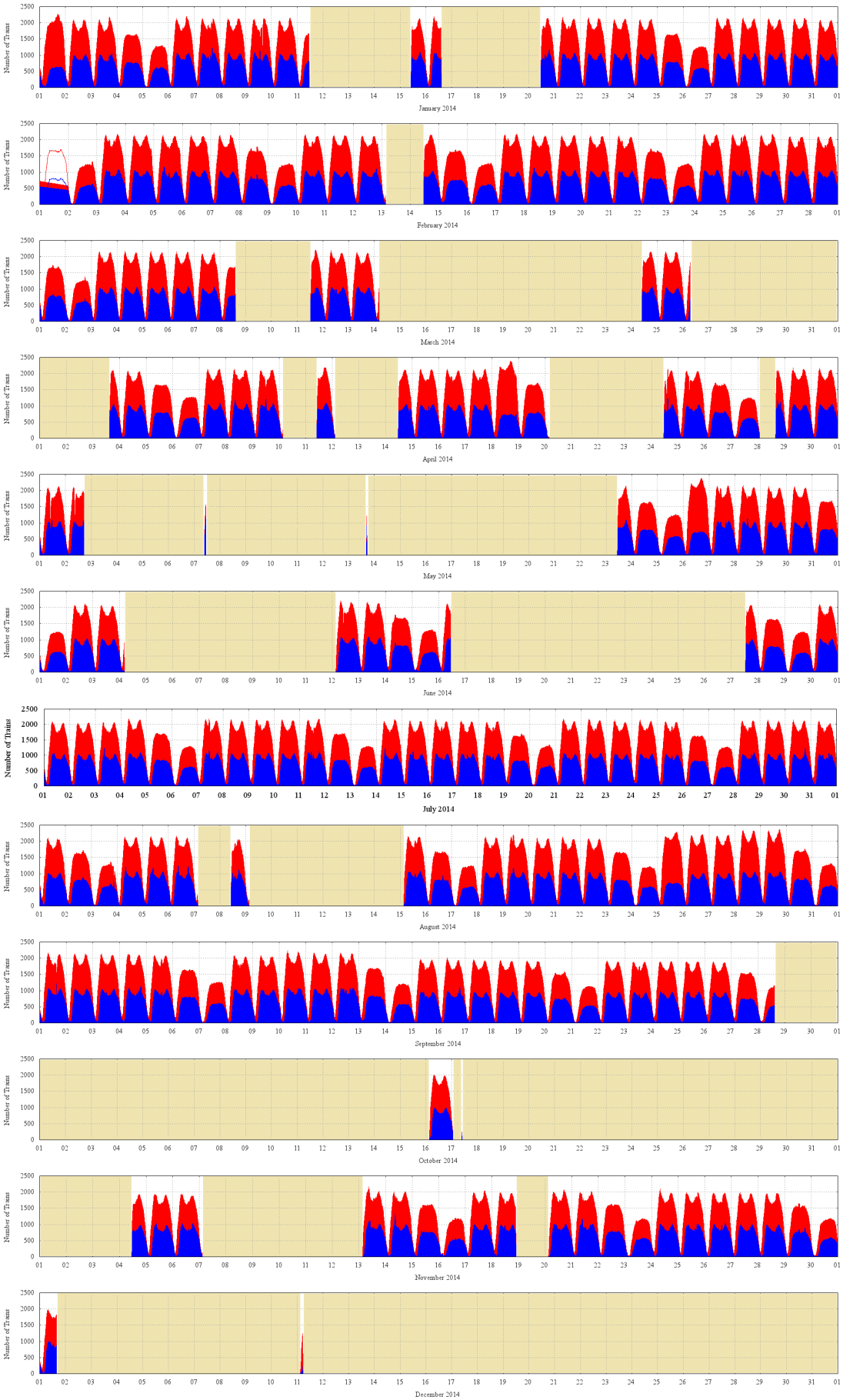


Figure 8.14: Network Rail Train Counts for 2014. The red plot is the number of trains in the whole of the UK, while the blue plot is only the trains within the London box (see text). Both are plotted from the y=0 axis so the count of actual trains can be read off directly. Yellow indicates missing data.

the year, where $22,997$ contain no data. This equates to approximately 43.63% loss. One other difference between this data and the tube and bus data is that it is based on 10 minute samples, rather than the 3 minute samples for the other two transport sources. This is largely for historical reasons, as, before the release of the Network Rail API stream in 2013, the data was obtained from the National Rail website for live information, which could only be queried at 10 minute intervals. As a result of these sampling differences, the rail data is only synchronised with the bus and tube twice an hour, although all the messages on the stream are retained to allow all the data to be reprocessed at a different sampling interval if required.

As the Network Rail real-time data also contains runlink lengths and number of late minutes, it is also possible to calculate a distribution for average time between stations and use this for a comparison with the bus and tube networks. Using the 'late minutes' data, figure 8.15 shows average late minutes per train plotted in real-time.

Figure 8.15: Network Rail, 10 May 2013 at 09:42. The magnitude of the plot for each train operator shows average late minutes per train. The graphic is taken from a website designed to show the information in real-time: *http://loggerhead.casa.ucl.ac.uk/visualisations/networkrail.html*.

This view of train delays is taken from the ANTS project, which provides a live web page showing the information in real-time. Average late minutes per train is plotted for all the train operating companies running trains through London. The result is not unlike a 'seismograph' for trains, indicating when problems are being experienced and is a useful feature detector for the Network Rail data. However, this is not the complete picture as number of cancelled services needs to be taken into account (snow days and 'leaf-fall' timetables), along with the runlink time. A service running 1 minute late on a run of 20 minutes is less significant than on a run of 5 minutes. Factoring in the percentage delay by total trip time is one method that has been tried, but no single statistic captures all the available information. Expanding this out to the full year, figure 8.16 shows the number of late running Network Rail trains in 2014 for all the train operating companies in the UK.

8.3.1 Interactions and Systems

Without the essential information about commuters, there are no interactions between commuter agents and tube, bus or train agents, which is a basic requirement for agent based modelling. One feature that comes out of the data is that lateness in trains, tubes and buses follows the underlying morning and evening commuter peaks, suggesting that large numbers of commuters put pressure on the system during these times. This is based on it taking longer to get on and off of a packed train and tube and rail companies build this information into their engineering timetables with additional waiting minutes. The aim of building this transport model is to enable experimentation based on real city data, so simulations of tube movements with rush hour waiting times built into the model can be compared to real life data.

Two interactions which are present in the current data are with air quality and weather, both of which should be linked to mass transport systems. If it is raining, more people could be expected to take the tube than walk or cycle. Buses running on petrol or diesel are a source of air pollution and weather is a major factor affecting air quality. This forms a complex three way system which can be investigated.

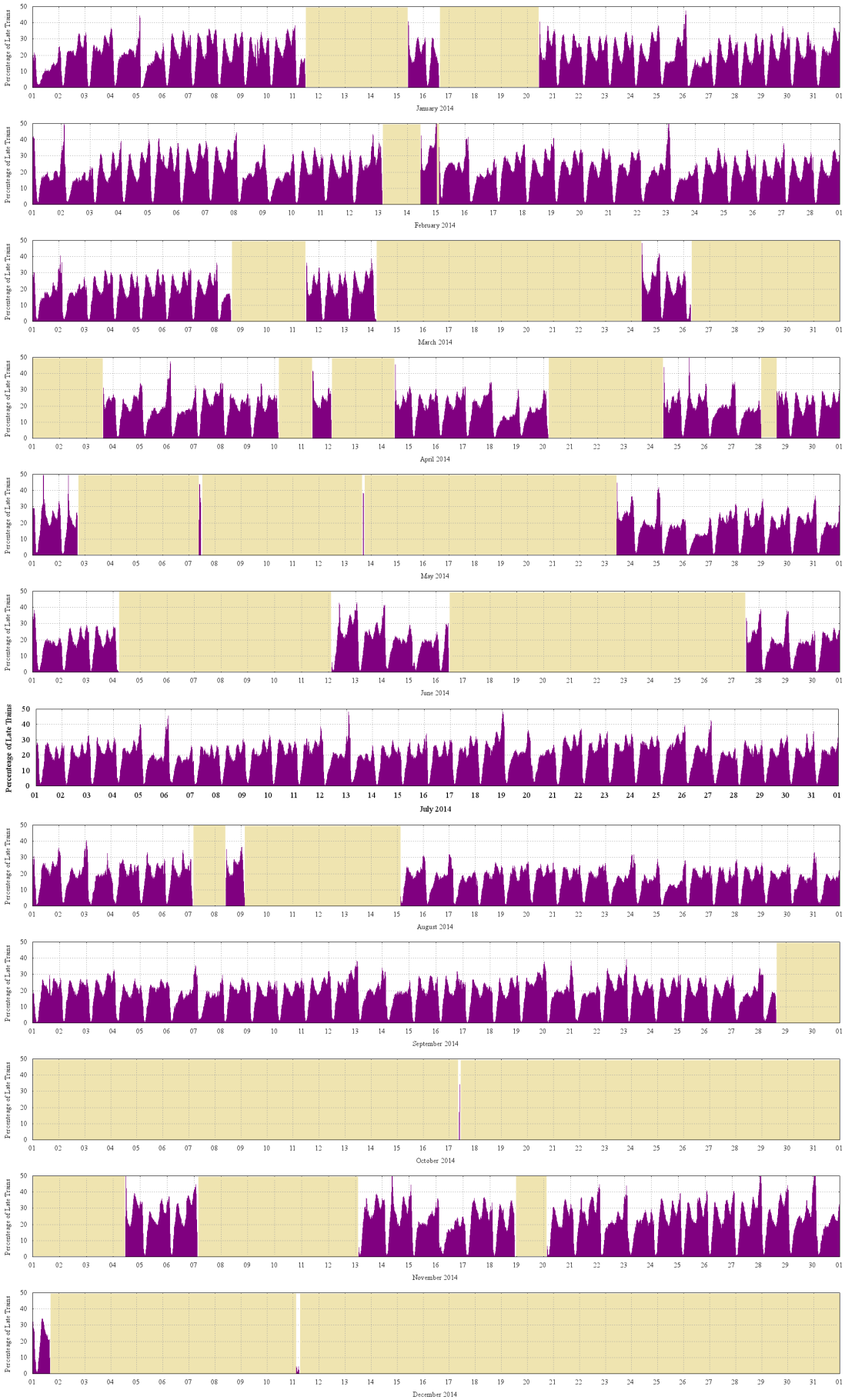


Figure 8.16: Number of late Network Rail trains as a percentage of the total number of trains running. Yellow indicates missing data.

Date	Description
2012	
Friday 22 June 2012	Bus Strike and earliest archive of bus data from TfL's new public 'Countdown' API
Friday 27 July 2012	London Olympics opening ceremony
Sunday 12 August 2012	London Olympics closing ceremony
2013	
Thursday 17 Jan 2013 to Tuesday 22 Jan 2013	Heavy snow causes problems for arrivals and departures at Heathrow
Friday 5 April 2013	TfL added river services to the bus API
Monday 22 April 2013	Lost previous 11 days of bus data due to archive failure
Saturday 7 September 2013	A TfL change to the Countdown API results in the CityDBAPI service's IP being blocked due to excessive usage and data loss over the weekend
2014	
Wednesday 5 February 2014	Tube strike
Thursday 6 February 2014	Tube strike day 2
Tuesday 29 April 2014	Day 1 of tube strike (from 9pm last night)
Wednesday 30 April 2014	Day 2 of tube strike
Wednesday 11 June 2014	Taxi strike at 2pm
Friday 12 December 2014	NATS failure caused the closure of UK airspace for 36 minutes between 15:27 and 16:03.
2015	
Tuesday 13 January 2015	Bus strike
Thursday 5 February 2015	Bus strike, second day of action, first was in January
Wednesday 8 July 2015	Tube strike starts at 5pm (approx) and tomorrow
Thursday 9 July 2015	Tube strike day 2

Table 8.2: Notable London events from 2012 to 2015.

8.4 Environment

As the 2014 data archive contains all of the air quality data from the AURN network, along with weather data from the Meteorological Office's 'Datapoint' website, an analysis of bus and tube data and their effect on air quality is possible. While tube data might seem strange, there are 4 tube strike days during 2014 when TfL had to run more buses than a normal day. As these were extra bus services, using any vehicles available in addition to the regular bus fleet, a large proportion did not have the 'iBus' tracking system on them and so do not appear in the bus data. In addition to tube strikes there are 3 bus strike days, one in 2012 and two in 2015, so data exists for fewer tubes and more buses, along with fewer buses and greater tube and bike use. Along with bus and tube strikes, a failure of the National Air Traffic Control System's (NATS) control computer in Swanwick caused the shut down of UK airspace in December 2014. All of these unusual events have the effect of removing one variable from the analysis, but their infrequent nature and dependence on other variables makes obtaining any definitive results difficult. Showing that air quality in London improved dramatically on the day of a bus strike would be an interesting result, but it could be the result of meteorological conditions. Given the small number of bus and tube strikes which have happened, a more realistic proposition is to partition the data into two classes based on the following: $\{weekday\}$, $\{BankHoliday, Saturday, Sunday\}$. Table 8.2 contains a list of the most notable events affecting the London data collected from June 2012 until August 2015.

In order to perform this analysis, the weather data needs to be examined in conjunction with air quality. There are 148 air quality monitoring sites in the UK which are part of the AURN network[4]. These are listed in the table in appendix D. Data from the UK Meteorological Office is available through their 'Datapoint' API and contains 135 surface observation stations. These are listed in appendix E. While other sources of air quality and weather data are available, which might offer improved temporal and spatial coverage, the two official sources of data have been used initially as both have rigorous quality control.

Figure 8.17 shows the locations of AURN and Met Office sites in London with a UK view in figure 8.18. Of these, only 18 air quality sites are within the London area (HIL, HRL, TED2, TED, HR3, EA8, KC1, CA1, MY1, HORS, CLL2, HG4, HG1, SK5,

[4]There are 151 listed in the appendix as it contains 3 which closed in 2014.

TH2, LON6, BEX, THUR), while the only weather observing stations near London are Heathrow, Northolt, Kenley and Gravesend-Broadness. These sites were determined using the London box defined for the Network Rail data earlier, based on the following extents: $((-0.51, 51.286), (0.34, 51.692))$.

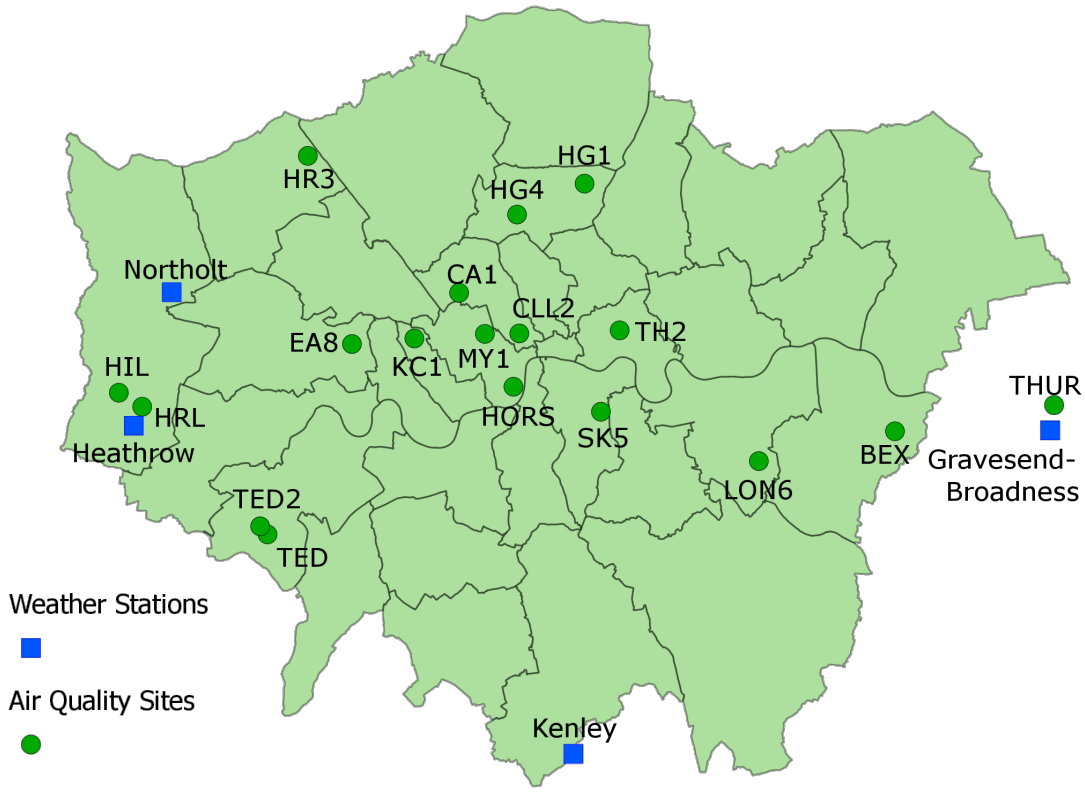


Figure 8.17: Met Office Datapoint stations (blue squares) and AURN air quality monitoring stations (green circles) in the London area.

As can be seen from the map, two air quality monitoring stations are situated close to Heathrow. These have the site identifiers of 'HRL' (Harlington) and 'HIL' (Hillingdon). From the AURN data, the measurements available are: sulphur dioxide (SO_2), nitrogen dioxide (NO_2), ozone (O_3) and particles $PM10$ and $PM2.5$. The units of measurement and methodology for each is shown in the following table:

Name	Method	Units
Ozone (O_3)	Running 8 hour mean	μgm^{-3}
Nitrogen Dioxide (NO_2)	Hourly mean	μgm^{-3}
Sulphur Dioxide (SO_2)	Maximum 15 minute mean	μgm^{-3}
PM2.5	Running 24 hour mean	μgm^{-3}
PM10	Running 24 hour mean	μgm^{-3}

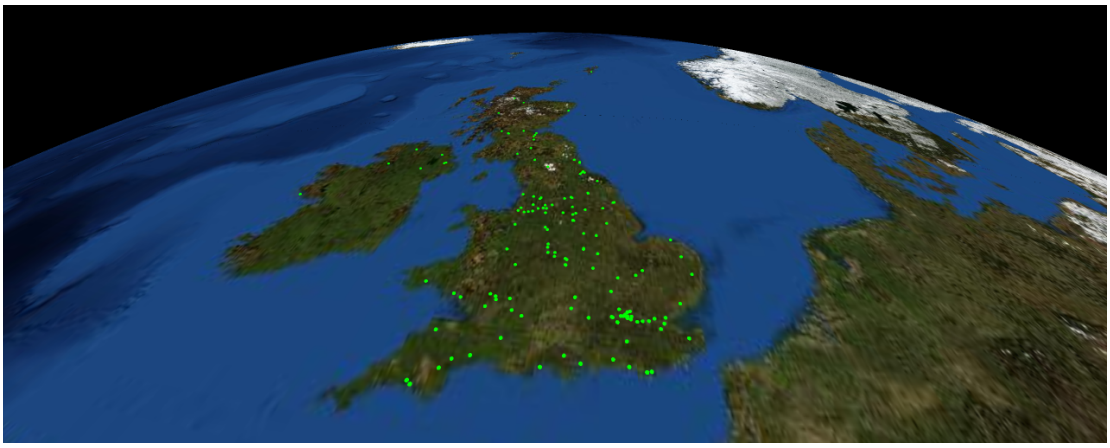


Figure 8.18: 148 AURN Air Quality monitoring sites in the UK.

Nitrogen dioxide (NO_2) emissions are a by-product of diesel engine combustion, so this variable is of particular interest with bus numbers in mind, but the first analysis is to look at the effect of weather on the air quality data. Wind speed and direction is the dominant factor, with air temperature and mean sea level pressure also secondary factors ("Mapping Carbon Monoxide Using GPS Tracked Sensors", [MS07]).

Using the Datapoint archive and Air Quality archive for 2014, a model was created using the GeoGL program to load data from each source, synchronise the times and write out a log file using the following format:

```
timecode,wxname,wmo,reportutc,WxType,Visibility,AirTemp,
WindSpeed,WindDir,WindGust,MSLP,airqname,O3,NO2,SO2,
PM2p5,PM10
```

An example of the data for Heathrow is as follows:

name	value	name	value
timecode	20140101_000000	WindGust	0
wxname	Heathrow	MSLP	1002
wmo	03772	airqname	HRL
reportutc	01/01/2014 00:00:00	O3	26
WxType	7	NO2	33
Visibility	19000	SO2	0
AirTemp	7.7	PM2p5	6
WindSpeed	10	PM10	11
WindDir	180		

Using the data in this format makes it possible to use a variety of 3rd party tools for further analysis. First, though, the number of missing data values were calculated. For the weather data, only 5112 hourly files were present out of a total of $24 \times 365 = 8760$ for the year, which equates to 58.4% of the total data available being archived. The reason for this high data loss can be traced back to a problem with the Datapoint API. Around July and September, changes were made to the system which resulted in problems with files not being available for download. After making a "GetCapabilities" request to establish the existence of a new file to download, the file could not actually be downloaded until several hours later. While the 9am file might show as available, all attempts to download it fail until some time later in the afternoon. This issue and other API failures have been discussed at length on the Datapoint Google Group[5], but no solution has been forthcoming. While this might seem like a big issue, other sources of weather information are available, as all Met Office synoptic observations are archived and can be downloaded in bulk from the British Atmospheric Data Centre (BADC) or MIDAS archive. Another source is the Weather Underground site, which also archives the official Met Office data in the form of Synops and Metars from a GTS feed rather than using Datapoint. The statistics on missing data for each month are shown in table 8.3.

	Jan	Feb	Mar	Apr	May	Jun
Hours Missing	79	89	79	49	15	152
Hours in Month	744	672	744	720	744	720
Percent	10.62%	13.24%	10.62%	6.81%	2.02%	21.11%
	Jul	Aug	Sep	Oct	Nov	Dec
Hours Missing	440	430	382	584	667	683
Hours in Month	744	744	720	744	720	744
Percent	59.14%	57.80%	53.06%	78.49%	92.64%	91.80%

Table 8.3: Missing weather data hours for 2014.

By replacing the September, October, November and December data with information from one of the public archives, the hourly coverage is increased to 83.98% with 7356 hours of valid synoptic observations and 1430 hours of missing data. An additional column has been added to the data containing the number of hours since the start of the year as a safeguard against any systematic error being introduced as a result of using weather observations from two different sources. The data used here are the official Met Office Metar reports, as archived and downloaded from the Weather Un-

[5]Met Office Datapoint Google Group: metoffice-datapoint@googlegroups.com.

derground website[6], but the 'navlost.eu' site also archives Metars as far back as 2008[7]. In this case, the Metar reports have already been decoded, so the data is already tabulated, but decoding software for a number of Meteorological formats is available from the MapTube GitHub repository: *https://github.com/maptube/WxDecoder*. Although the reports have been decoded, a significant amount of pre-processing is required as the units are in kilometres per hour and kilometres, where the original data used miles per hour and metres. The Metar reports are also missing the weather type and pressure tendency.

The inclusion of an 'hour of day' field in the data allows the daily variation pattern to be plotted. If NO_2 is produced by bus and other road traffic, then a daily variation which follows established commuter behaviour is an expected result. By generating hourly averages for all the AURN sites within the London box (figure 8.17), the daily variation patterns of NO_2 can be extracted to test this hypothesis. The graphs in figures 8.19, 8.20 and 8.21 show the daily variation for every site, calculated from all of the available 2014 data. The graphs have been plotted with the same Y scale to allow easy comparison of absolute NO_2 levels between sites, although this does flatten the graphs with lower peak values and mask the morning and evening peaks. One interesting results concerns the Marylebone Road (MY1) site, where there is a linear increase from morning to evening peak rather than a drop during the middle of the day. The sites 'EA8', 'HR3' and 'TED2' have been included for completeness, but the data shows that they do not have the sensors to record NO_2. The data in figures 8.19, 8.20 and 8.21 is an important result as it clearly shows NO_2 levels exhibiting a clear morning and evening peak which is indicative of the commuter patterns of tube, bus and train numbers explored earlier in the chapter. The same signal is present in this data as the other transport data, adding weight to the argument that these are interacting systems.

In order to interpret the information for each air quality site displayed in figures 8.19, 8.20 and 8.21, it would be useful to be able to include traffic counts. Unfortunately, while the Highways Agency data is available via the *data.gov.uk* data store every 20 minutes, TfL is responsible for traffic flow in the London area and the format of data is different from the rest of the country. The Highways Agency data contains counts from traffic sensors which give raw vehicle counts, average speed, vehicle type and journey time data. The TfL data only contains information about delays and delay data

[6]Weather Underground: *http://www.wunderground.com*.
[7]NavLost: *http://www.navlost.eu/aero/metar/*.

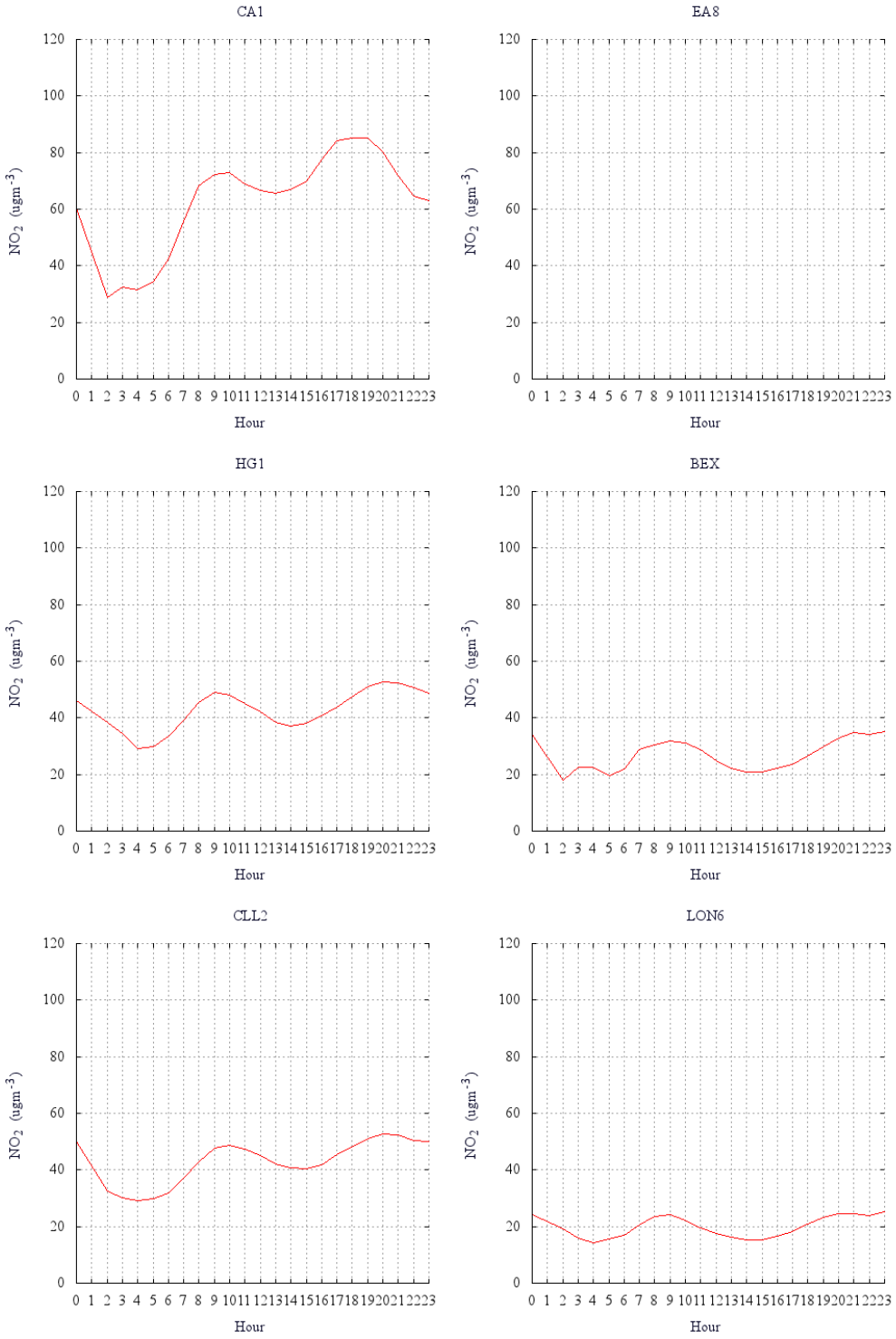


Figure 8.19: NO_2 levels for sites within London averaged over all of 2014. All graphs use the same scale. A morning and evening peak is clearly visible in most of the charts, indicating a link with human activity as this effect cannot be explained by environmental factors. Page 1/3.

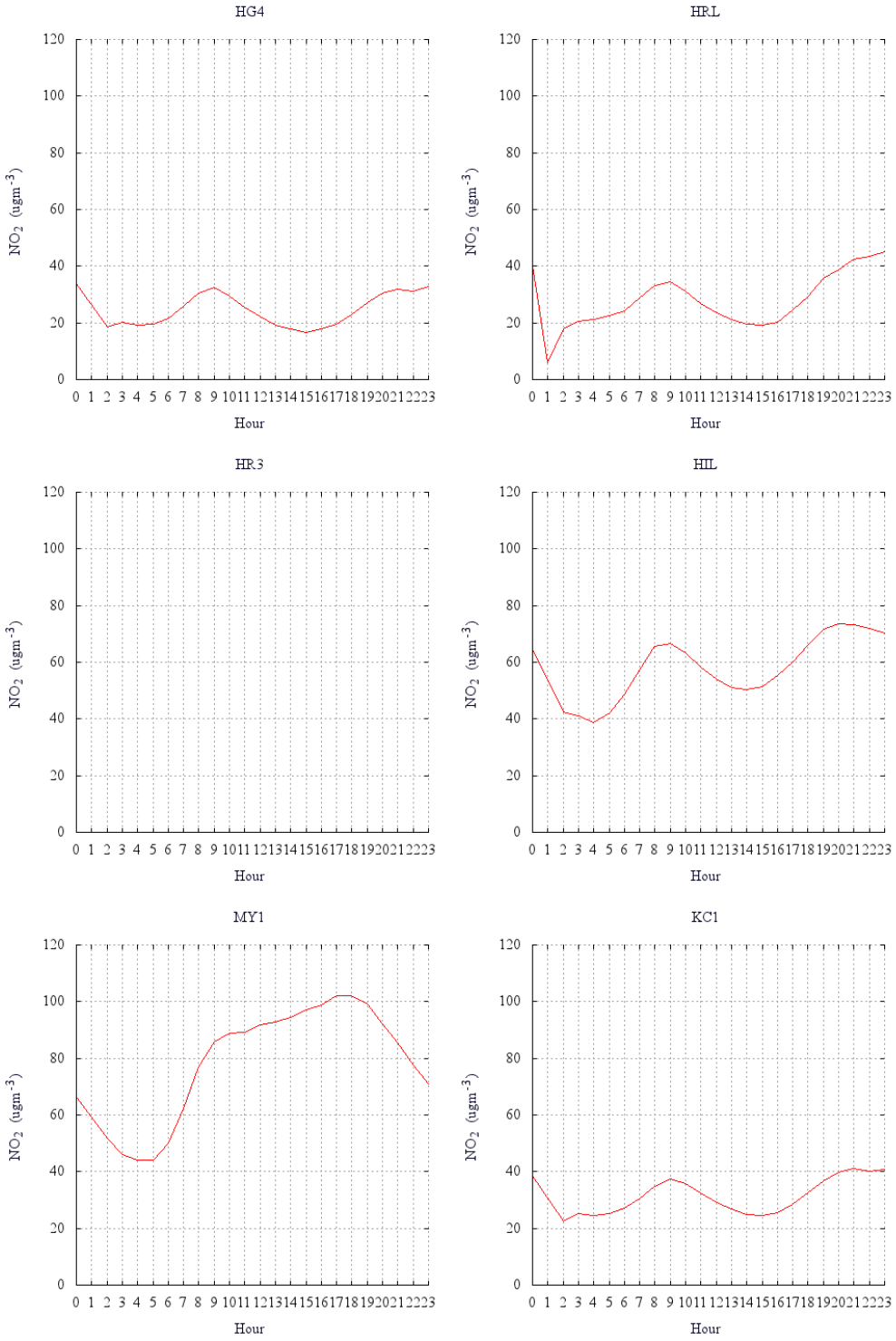


Figure 8.20: NO_2 levels for sites within London averaged over all of 2014. All graphs use the same scale. A morning and evening peak is clearly visible in most of the charts, indicating a link with human activity as this effect cannot be explained by environmental factors. Page 2/3.

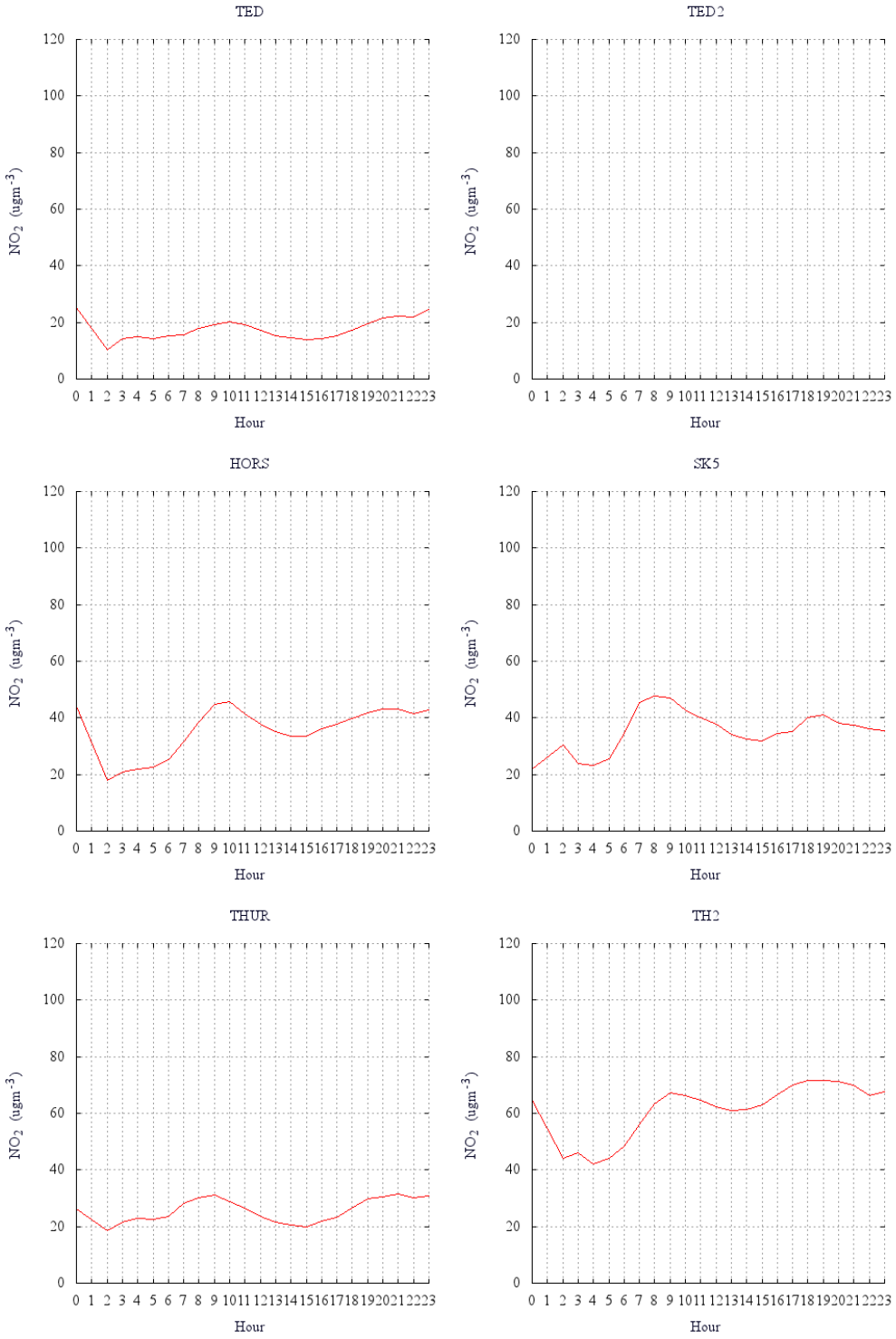
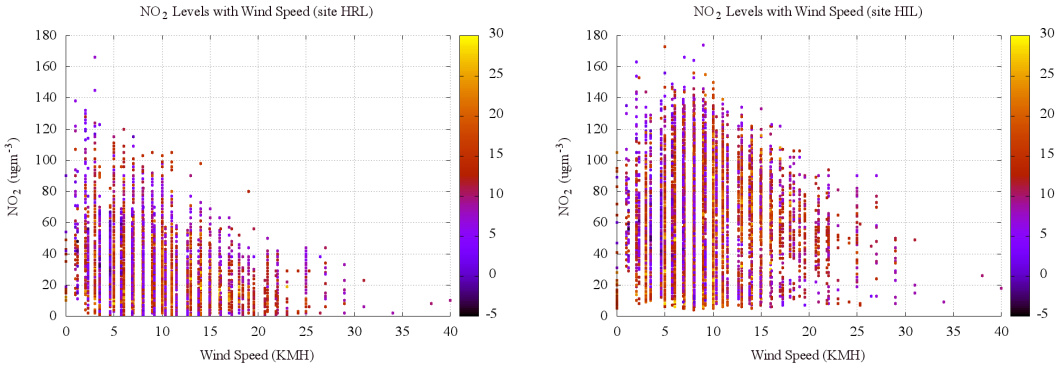


Figure 8.21: NO_2 levels for sites within London averaged over all of 2014. All graphs use the same scale. A morning and evening peak is clearly visible in most of the charts, indicating a link with human activity as this effect cannot be explained by environmental factors. Page 3/3.

from matrix warning signs is of no use in estimating traffic flow. Other traffic sensor networks do exist which cover the London area, for example Google's traffic map based on GPS travel times from cars, but no open source of data exists at present.

Looking at the data for the Harlington site with the Heathrow weather data, figure 8.22a shows the NO_2 levels plotted against wind speed as a scatter graph with the points coloured according to air temperature. A similar result is obtained when the data for the Hillingdon site is plotted in figure 8.22b. These graphs both show NO_2 levels decreasing with an increase in wind speed which is to be expected for a gas being dispersed in air.



(a) Harlington. (b) Hillingdon.

Figure 8.22: NO_2 levels against wind speed and colour coded according to air temperature for the Harlington and Hillingdon air quality sites.

Another interesting result is shown in figure 8.23 where PM2.5 is plotted against PM10. The graph shows a clear linear dependency and a link between PM2.5 production and PM10.

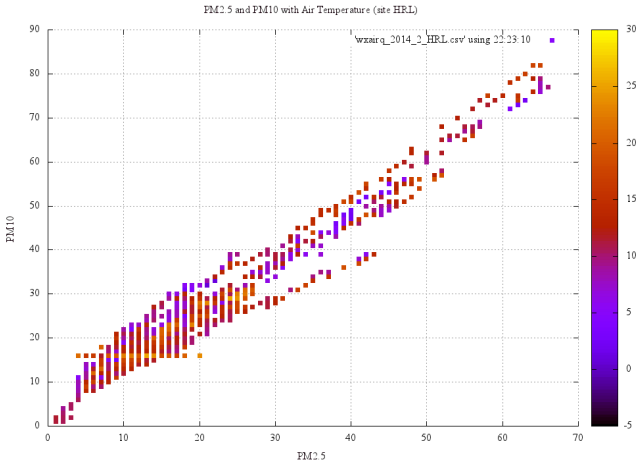
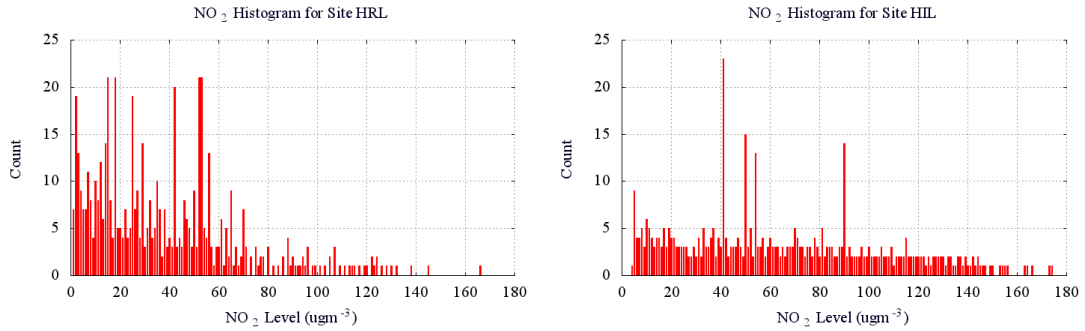


Figure 8.23: PM2.5 plotted against PM10 and colour coded according to air temperature for the Harlington air quality site.

While NO_2 levels can be shown to decrease with wind speed, it should be recognised that calm conditions are more frequent than extreme weather events, so the number of data points for the higher wind speeds become sparse. These effects are visible in the frequency histograms shown in figures 8.24a, 8.24b and 8.25, which show the frequency distribution of NO_2 levels for Harlington (HRL) and Hillingdon (HIL), along with wind speeds for Heathrow.



(a) Harlington NO_2 distribution.

(b) Hillingdon NO_2 distribution.

Figure 8.24: NO_2 frequency histogram for the Harlington air quality site. The Y value shows the count of the number of times a particular NO_2 value (x-axis) was recorded during 2014.

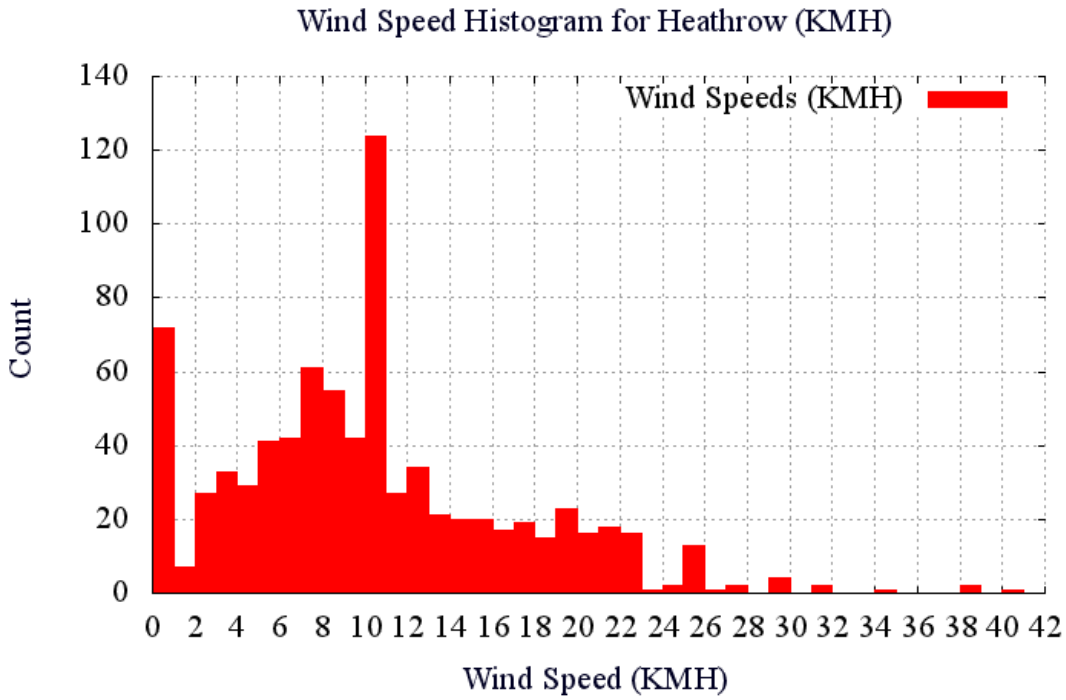


Figure 8.25: Wind speed frequency histogram for Heathrow. The Y value shows the count of the number of times a particular wind speed (x-axis) was observed during 2014.

The frequency plots for NO_2 are based on the AURN hourly averages of the data, so the frequency plot only shows counts of these averages, rather than being based on the raw data. One feature which is immediately apparent is how closely the HRL NO_2 sensor tracks the HIL one. For completeness, the NO_2 levels and wind speeds on a month by month basis for the whole of the year 2014 are plotted in figures 8.26 and 8.27 respectively.

One additional factor which needs to be understood is that the air quality stations are designated as either 'roadside', or 'background', depending on whether they are situated in direct proximity to a major road or some distance away. This information is in the description data for every sensor, so, for example, the Marylebone Road sensor (MY1) is a roadside sensor, being situated directly at the edge of the kerb half way along Marylebone Road. This highlights an additional benefit of using the 3D model, as the position of any sensor can be seen in relation to the surrounding roads and buildings. The allometry of buildings in London and also in other countries has already been studied [Bat+08], but using a full 3D GIS to analyse the fabric that makes up a city in relation to other data, including real-time sources, is only just starting to become technically feasible. Building data could be included along with the air quality data as building layout around sensors has a direct effect on the data recorded. If building heights were made open data and added to the 3D model, then a sensitivity analysis of the effect on wind direction around a sensor based on street layout would be possible. Without going to the complexity of a full computational fluid dynamics model, a basic analysis of street canyons from their width to height ratios and which roads would be expected to have higher levels of pollution based on wind direction is all possible.

In light of this traffic flow data hole covering London, the next alternative is to use the bus data as a proxy for traffic flow. However, while buses and cars both use the road, buses have dedicated bus lanes and priority over cars at selected traffic lights and junctions. Bicycles, motorbikes and black cabs are permitted to use bus lanes too, so there is the potential that the slower bicycles could be affecting bus flow to a greater extent than the other traffic. In "Using Bus Probe Data for Analysis of Travel Time Variability" [Uno+09], the authors use data from bus travel time to assess the travel time variability and level of service of roads. They make the point that, "one of the advantages of using Bus Probe Data is that it can easily provide data on fluctuations in travel time because the buses travel repeatedly along the same fixed routes". They then

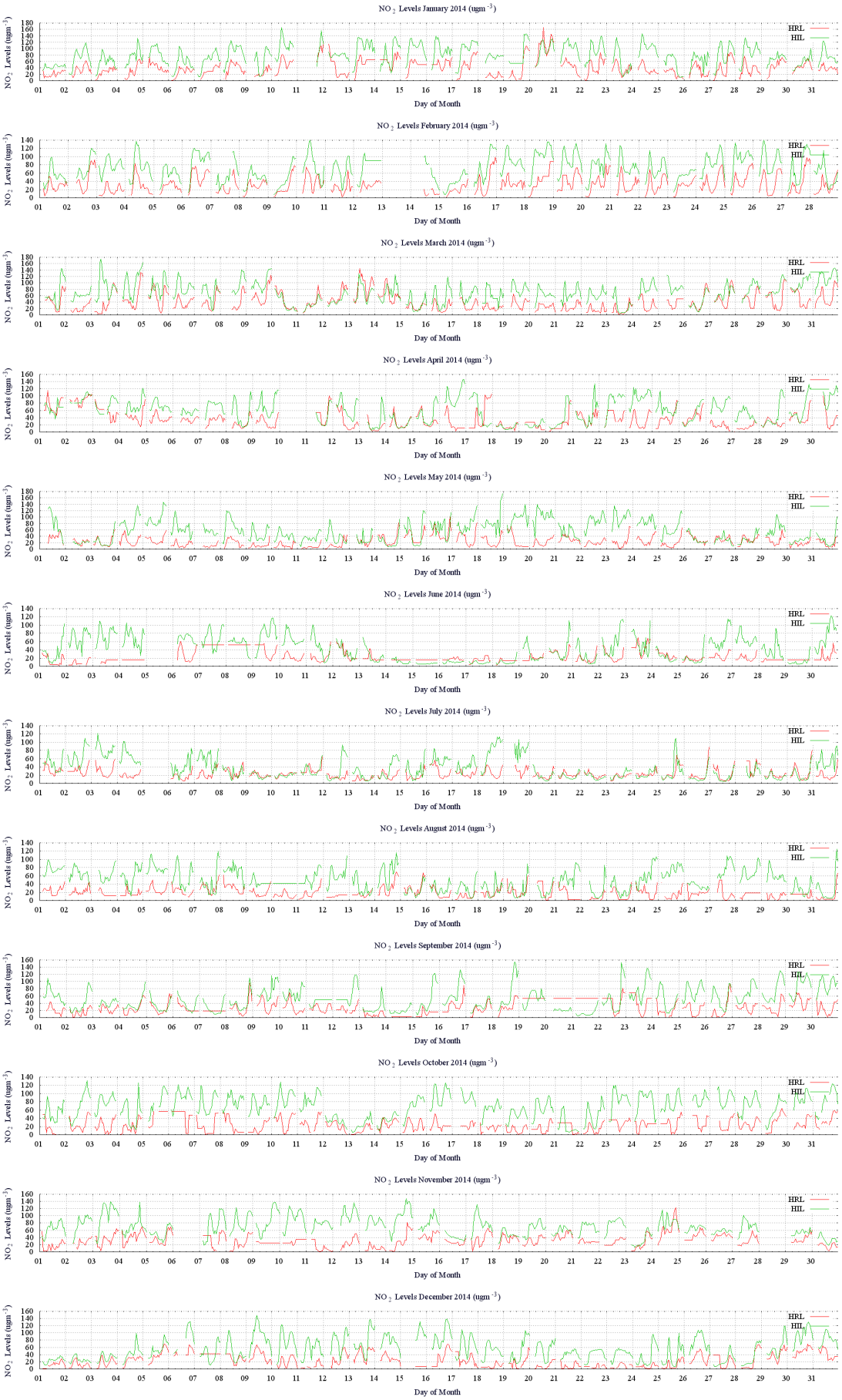


Figure 8.26: NO_2 levels for the HRL and HIL sites for 2014.

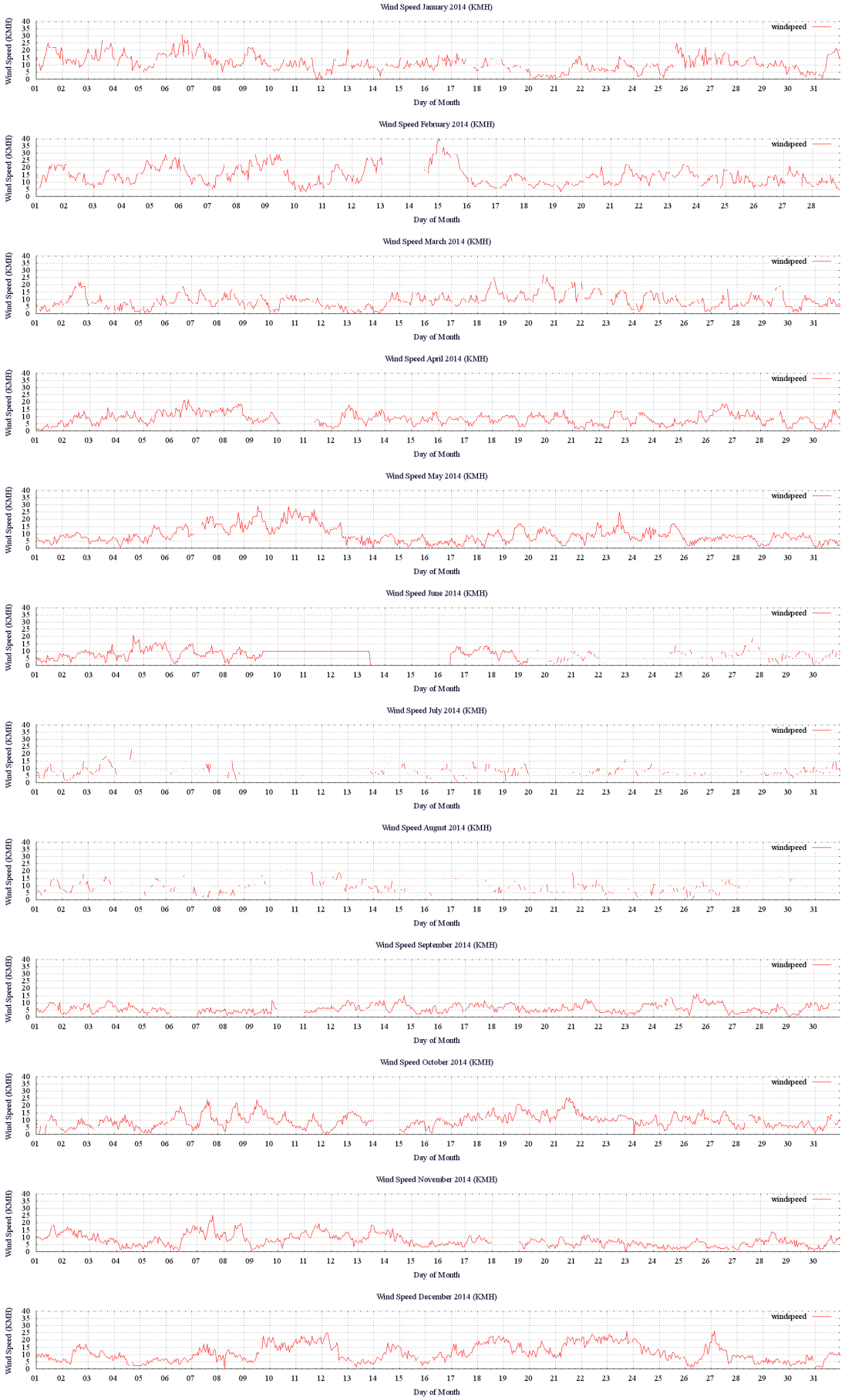


Figure 8.27: Wind speeds recorded at Heathrow for 2014.

go on to point out the disadvantage, which is that they "overestimate travel time due to the time spent at bus stops". In the main part of their paper they develop a model of road level of service, comparing GPS tracked bus data with GPS data from a car driven along the same route. Their results are encouraging and suggest that using bus data as a proxy for traffic is a viable methodology.

By using the bus model that was created earlier, along with the position of an air quality sensor, it is possible to add an instrument to the model that counts how many buses are in the vicinity of the sensor. There are a number of ways of doing this, for example, geo-fencing an area around the sensor and counting the numbers of bus agents that enter, or using an inverse distance weighting function to calculate an agent density for a point in space. The key issue here is to use the model of bus movement to allow experimentation on the real city data. This is differs from the more conventional approach of analysis directly from the raw data, as here, the bus movement model is being used to interpolate positions between the 3 minute sample points to give a finer temporal resolution to the data. While this could, in practice, be achieved directly from the raw data files and some additional programming to determine if movement vectors along the graph edges cut the geo-fenced box, the approach taken here is intended to have a wider range of application. By using a bus model which was trained using real data, the creation of new instruments to ask "what if" questions of the data in an ad-hoc way is greatly simplified. Then, by running multiple models simultaneously, interactions between them can be investigated. This example concerns air quality and road traffic, but something which could also be investigated is the street layout and wind direction and how this also affects air quality. As shown in [MS07], wind direction along street canyons or perpendicular to the street layout has a bearing on air quality, which is related to the width to height ratio of the street canyon. By using a 3D model, this information is potentially available for analysis and visualisation. The only reason it has not been included in more detail here is the fact that the building height data is not open and so cannot be made public at this point in time.

Returning to the problem of measuring local bus density around an air quality sensor, computational speed now becomes an issue, as there are up to 7,200 bus agents moving around on a simulation of a network with 30,000 bus stop nodes. While it is possible to let the simulation run for the whole year, the amount of time taken is prohibitive. The main performance bottleneck is the loading of the data from disk for each 3 minute time

segment and the graph processing required to place the buses in the correct position. This results in an estimate of 21.7 hours to process the data for a month, based on an hour taking 105 seconds. However, with air quality data only available for every hour, a faster method is to jump the simulation to the point where each air quality measurement is made and ignore the rest of the time. Alternatively, if air quality data was available at greater temporal resolution, for example via a 3rd party sensor, then a limited segment of time could be analysed in more detail instead.

Taking the 'MY1' air quality site on Marylebone Road as an example, numbers of buses within a 200 metre area can be logged and this compared with the NO_2 levels from the sensor. The area around the Marylebone Road site, is enclosed with an axis aligned bounding box which serves as the geo-fenced counter. Figure 8.28 shows the bus data for January 2014 based on the count of the number of buses passing through the defined area for 10 minutes either side of the hour. While these numbers initially look higher than expected for the number of buses passing along a road, the algorithm counts the same agent multiple times if it stays within the box for more than one minute during the ten minute sampling window. While it is a simple task to only count unique agents as the licence number of the bus is used as the agent's name, the counter was intended to give a count which is representative of the number of buses in the vicinity of the sensor. Along with adding a unique agent counter, another possible alternative is to calculate density based on an inverse distance weighting function.

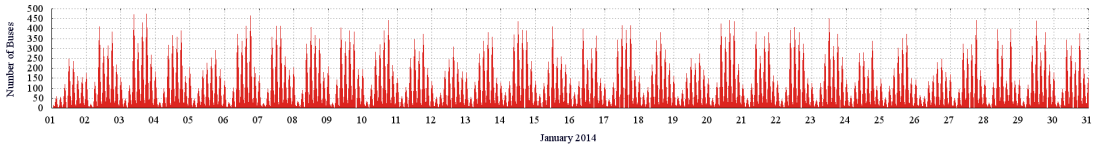


Figure 8.28: Number of buses passing through the MY1 site on Marylebone Road during January 2014.

The bus numbers show a similar daily variation pattern to figure 8.4, earlier, which showed the morning and evening peaks for all buses running in London. The task now is to examine the NO_2 data for days when bus strikes took place. While the first recorded strike during the period of the data was on Friday 22nd June 2012, this is before the air quality and weather data was available in their current format. However, two strikes took place in 2015, on Tuesday 13th January and Thursday 5th February. The data for January and February 2015 is shown in figures 8.29 and 8.30.

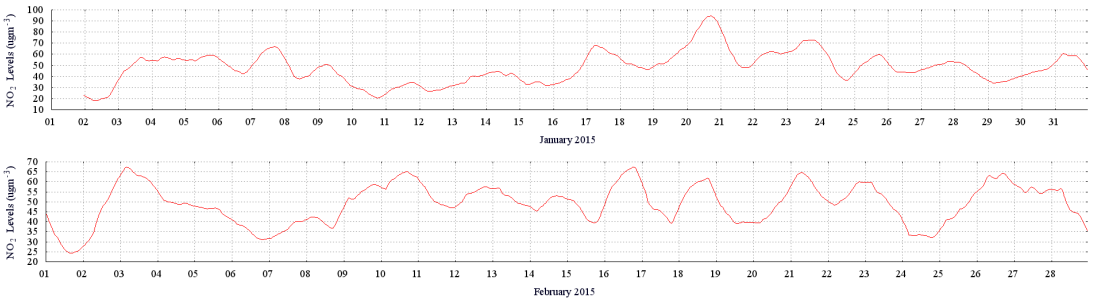


Figure 8.29: NO_2 levels averaged over all London sites and based on a 24 hour running mean for January and February 2015. Bus strikes took place on Tuesday 13th January and Thursday 5th February.

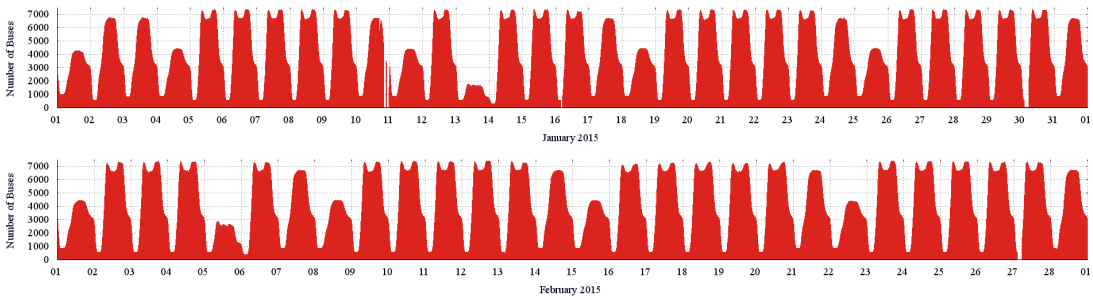


Figure 8.30: Bus numbers for January and February 2015. Bus strikes took place on Tuesday 13th January and Thursday 5th February.

This is compiled from all the air quality sites within the London box defined earlier for figure 8.17. The average across all sites has been calculated, then the plot shows a running 24 hour mean to remove the effects of the morning and evening rush hours.

The problem here is how to account for the effects of the weather and all the other traffic which was on the road in London as people tried to get to work without taking the bus. Using the analysis of rush hour times from earlier in the chapter, the peaks in bus numbers correspond to 09:00 (morning) and 17:00 or 18:00 (evening). As the morning peak was shown earlier to have a more well defined shape and as NO_2 levels increase throughout the day (figures 8.19, 8.20 and 8.21), the morning rush hour is the chosen option. The plots in figure 8.33 show the mean daily variation in NO_2, but this time only using data for Sundays and bank holidays. The reason for including Saturday in the weekday group is that the number of buses running on a Saturday is similar to during the week, but without the rush hour peaks. On a Sunday, there are roughly half the number of buses running[8]. The graphs for each London air quality sensor show a marked difference from before, with some showing a morning and evening peak, but all with reduced NO_2 levels. Following on from this, figures 8.31 and 8.32 show

[8]The data in figure 8.30 shows a Saturday peak of 6,800 compared to a weekday peak of 7,200 while Sunday is only 4,500.

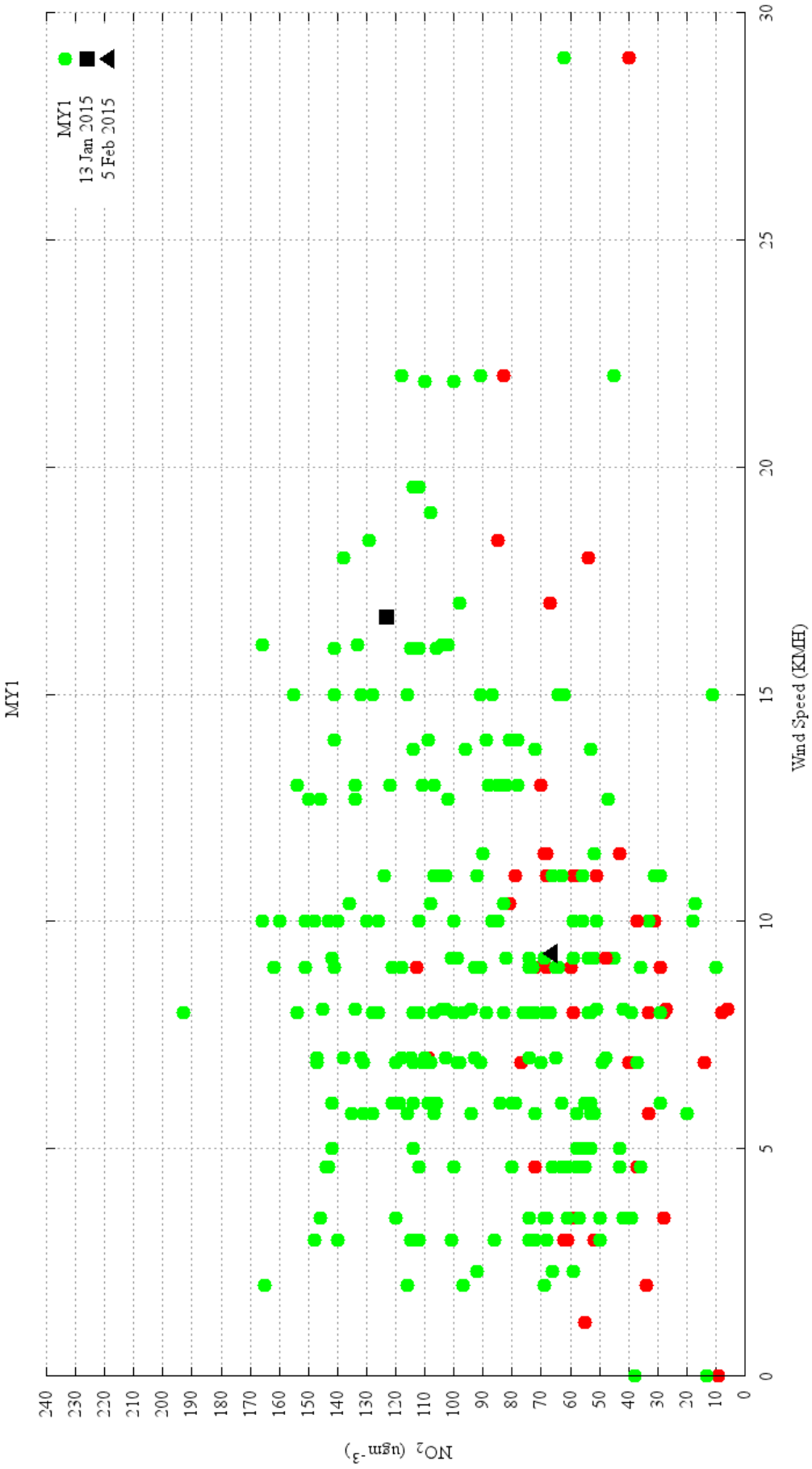


Figure 8.31: Marylebone Road sensor (MY1, roadside) NO_2 levels against Wind Speed (KMH) for 09:00 using the 2014 data. The colours represent Sunday or Bank Holiday (RED) and normal weekday plus Saturday (GREEN). Bus strike days on 13 January and 5 Feb 2015 are indicated with a black square and triangle respectively.

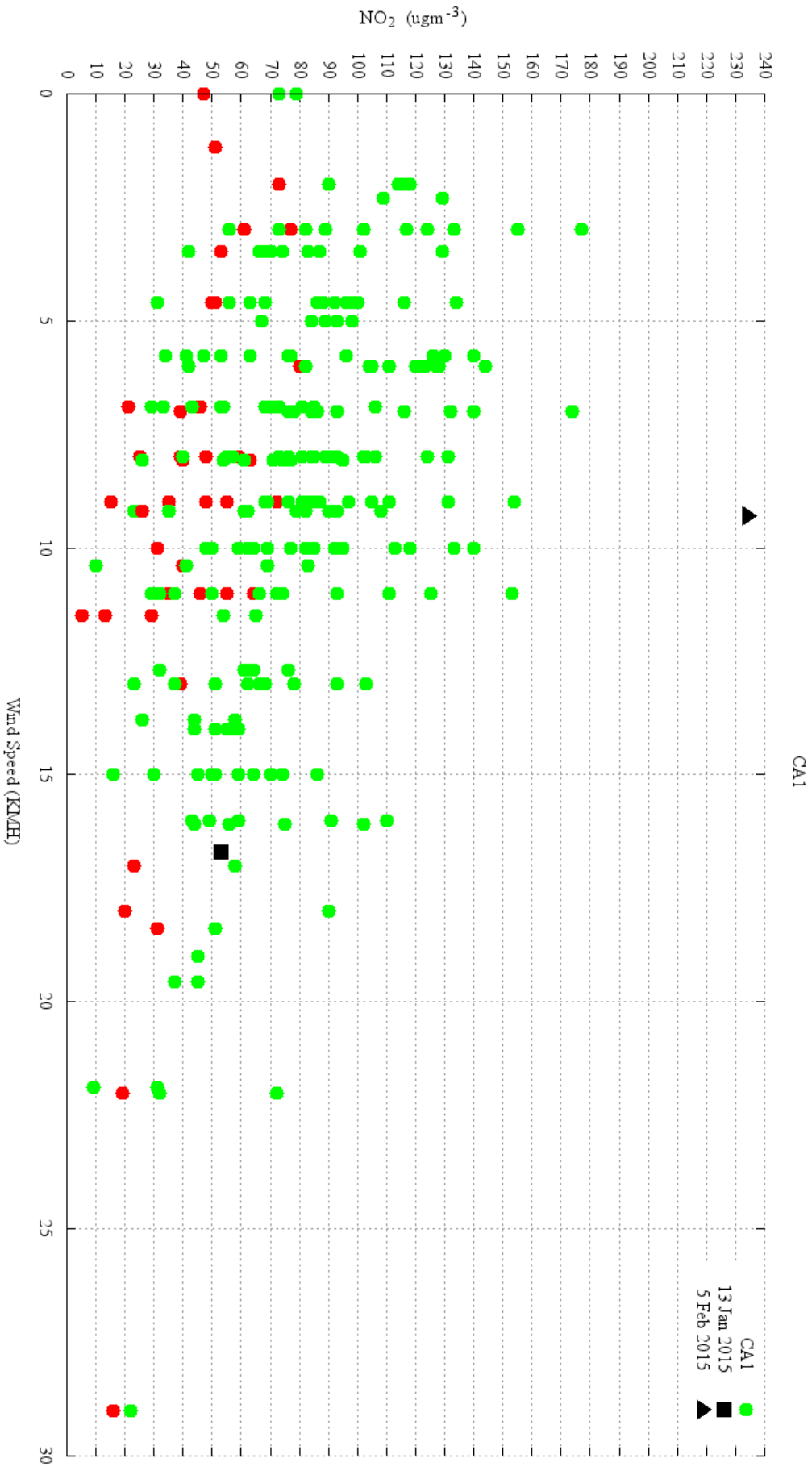


Figure 8.32: Camden sensor (CA1, roadside) NO_2 levels against Wind Speed (KMH) for 09:00 using the 2014 data. The colours represent Sunday or Bank Holiday (RED) and normal weekday plus Saturday (GREEN). Bus strike days on 13 January and 5 Feb 2015 are indicated with a black square and triangle respectively.

scatter plots of NO_2 against wind speed for two of the London sites: Marylebone Road (MY1) and Camden (CA1). The colour represents whether the data is for a Sunday or bank holiday (red), or normal working day and Saturday (green). In figure 8.31, the highest value of $190\mu gm^3$ is at 8KMH with the highest wind speed of 28KMH showing $60\mu gm^3$. The problem with this type of scatter plot is that the source of the pollution is not necessarily distributed evenly across all wind speeds, so figure 8.31 is missing the data at the lower wind speeds, which is present on figure 8.32. Despite the proliferation of points below 15 KMH in the first chart, joining up the highest NO_2 points for the 2, 8, 16, 22 and 28 KMH points reveals a definite decrease in maximum NO_2 levels with wind speed, even if the section up to 10KMH is flatter than the more defined profile shown in figure 8.32. The reduction in expected NO_2 levels as wind speed increases is evident, but both show lower NO_2 levels for the Sunday data. Plotting the data points from the bus strikes on 13 January and 5 February 2015 over the weekday data of figures 8.31 and 8.32 shows the difficulty in analysing this type of data. The 5 February data for CA1 would appear to be an outlier, with $234\mu gm^3$ NO_2 higher than expected. Without accurate data on numbers of private vehicles being used, though, it is impossible to draw any conclusions. This highlights the need for connected data analysis rather than analysing variables in isolation as a city is a connected system.

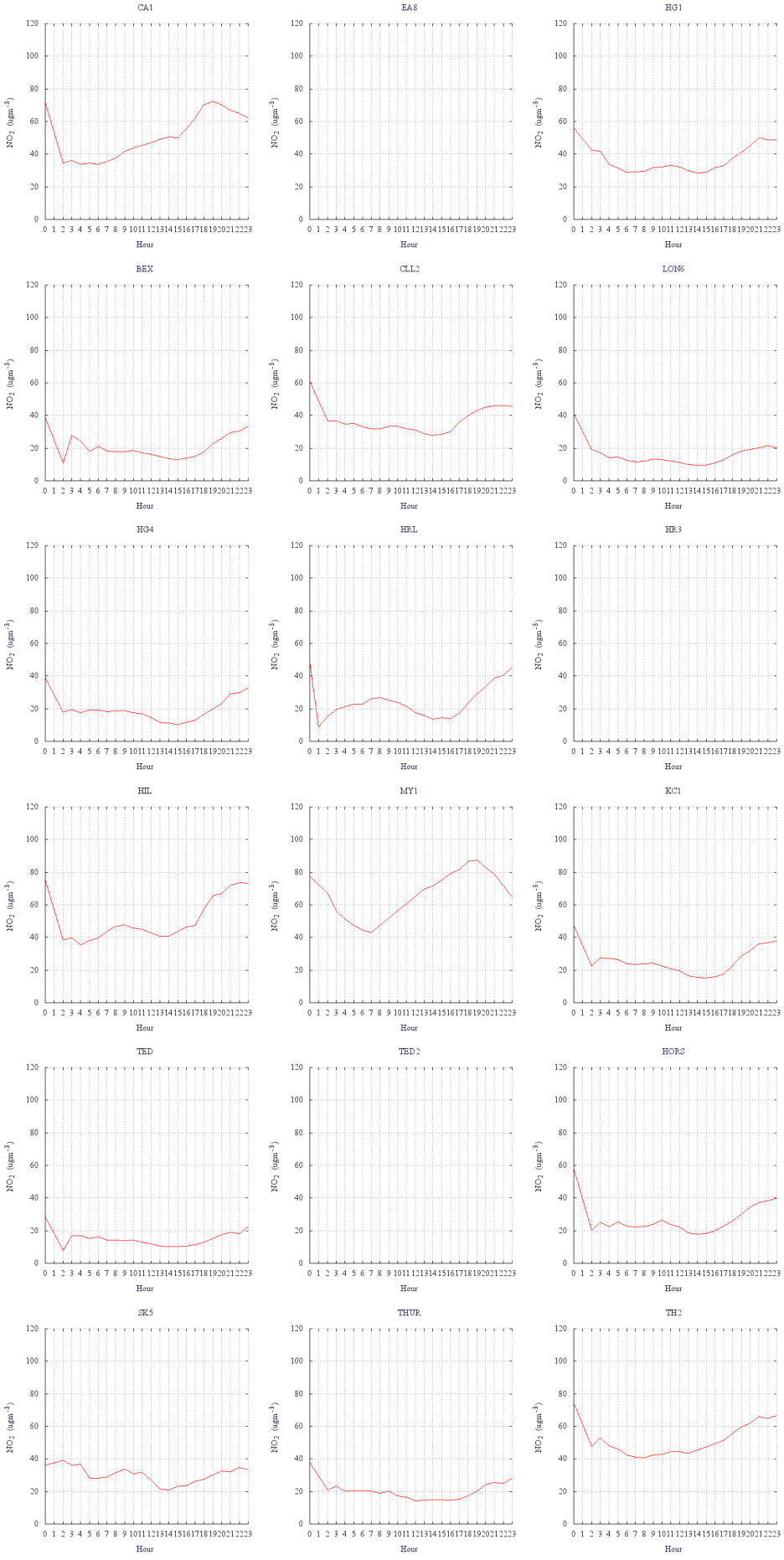


Figure 8.33: NO_2 levels for sites within London averaged over all of 2014, but only using data for Sundays. All graphs use the same scale.

8.5 Unified Data

This chapter started with the extraction of information from Internet data stores and the Census. These archives of data provide information about the static or slow to change structure of cities, for example through the population Census, travel to work data or other information stored in Internet data stores.

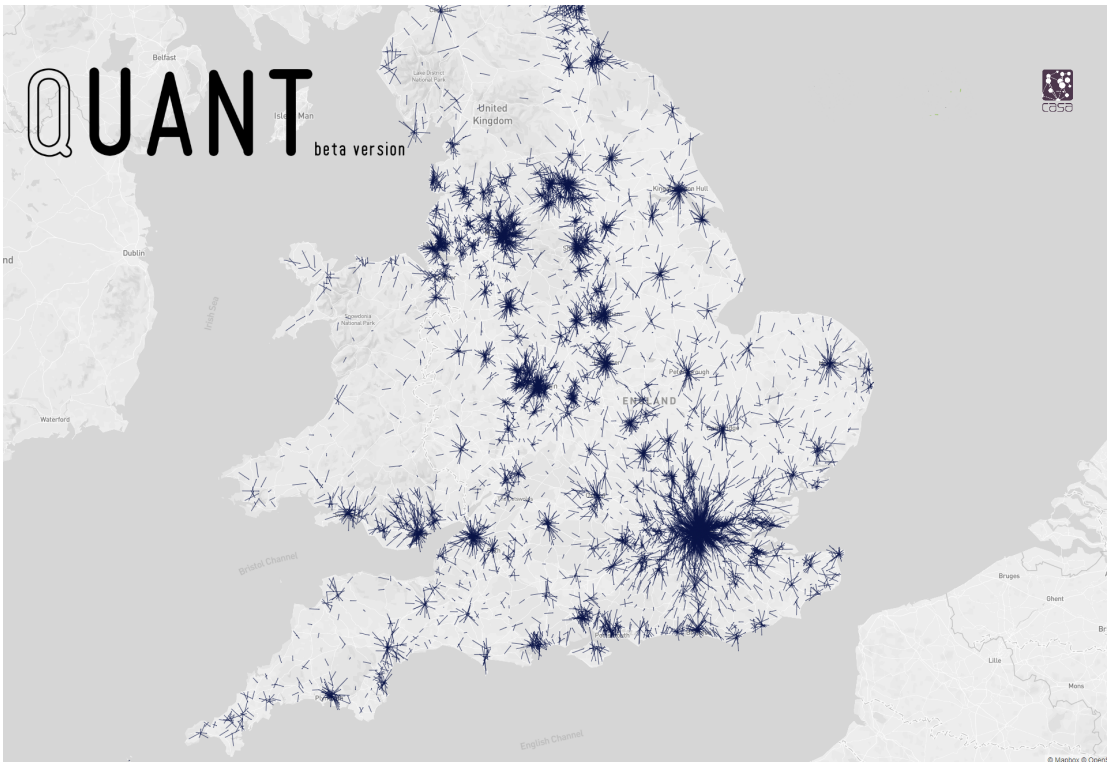


Figure 8.34: Commuter flows from the 2011 Census travel to work data.

While the map in figure 8.34 shows the commuting structure of England and Wales at the time of the 2011 Census, real-time information about commuter patterns is starting to become available which allows the exploration of the data at a finer spatial and temporal scale. Real-time sources of information use the Internet as a communications medium, both as a source for updating their information, and as a method for presenting the data to a wider audience in an easy to access and timely fashion. As the data becomes more complex, so do the visualisations and the method of presentation to the general public adapts to new technologies. The map shown in figure 8.34 is built from the Census travel to work data showing interactions between every combination of 'MSOA' areas[9] for all 7201 zones in England and Wales. The arrows show the mean direction of the greatest commuter flow from the origin zone to all others. This map

[9] MSOA stands for Middle level Super Output Areas.

uses 'WebGL' and works on an iPhone, opening up the possibility of more dynamic visualisations of real time data, delivered directly to mobile clients in a timely fashion.

The web services introduced at the beginning of the chapter to experiment on the data stored by the MapTube website also serve to feed data to the 3D visualisation system used to handle the real-time data. The river Thames shown in some of the visualisations is a GeoJSON file stored on MapTube and accessed via its unique URI. The buildings are extruded outlines from the OS OpenData release and accessed via another web service.[10] Similarly for UK roads and railways which make up the transport network, these are network graphs with geographically placed vertices. All of these are now seen as potential sources of information to be used in geographic knowledge discovery. The key is in how to connect all the systems and data formats together, which is the key reason for exploring data stores, real-time APIs and geospatial computing algorithms. A smart phone connected to the Internet now has access to a whole planet's worth of information. While the idea of a programmable planet might be a long way off, experimenting with real-time city data in conjunction with the extensive static data that now exists is a reality.

[10]The building outlines are not stored on MapTube because of the size of the data and the fact that MapTube is designed to store intangible data.

Chapter 9

Discussion and Further Work

Although Chapter 1 started with visualisation, there is nothing to visualise without data and the framework of web services built up into the ANTS project, which was introduced in section 4.5, has now provided over 4 terabytes of city scale data from 2013 to 2018. This is a valuable resource for researchers wanting to explore city systems and is currently being used in other academic projects. In addition to this, it also supplies bus, tube and cycle hire data for the real-time exhibit at the London Transport Museum (figure 9.1), while the origins of the real-time data acquisition project can be traced back to the iPad wall which was installed in the Mayor's office in January 2013 and the work done for the London Olympics in July 2012.

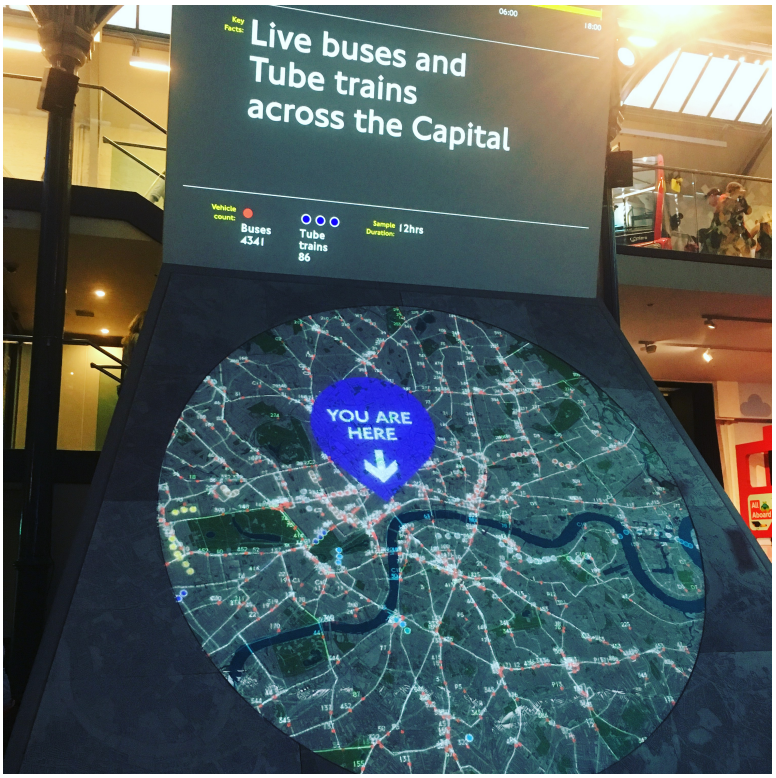


Figure 9.1: London Transport Museum live buses and tubes display.

The following sections are a discussion of how the research findings fit with the original research questions posed in section 1.2.

9.1 Discussion of Research Questions

Starting with the main research question, first presented in section 1.2:

> *Q1. How do automated mapping tools, real-time data and web-based mapping alter the landscape of geospatial analysis in an era of ever-increasing quantities of Internet data?*

Before this is broken down into its sub-questions, the points made regarding automated mapping tools, real-time data, web-based mapping and Internet data stores need to be covered. The web-based mapping is a technical question, but the point has been made earlier that video, word processing, spreadsheets and CAD can already be done in a web browser. Adding in Internet data stores provides a source of data not achievable on a desktop machine, as demonstrated by Google in their Fusion Tables service and ESRI in ArcGIS online. MapTube is a much smaller system than either of these, and so does not quite achieve the scalability of the commercial systems, but makes up for this by providing a different set of functionality. The aim of full exploratory spatial data analysis in a browser is still one step away, but the research in this thesis suggests that it is achievable in the short term. Chapter 5 looked at storage, visualisation and map comparisons, following on from the infrastructure analysis of the previous chapter, where formulas to predict performance were derived. In the literature review, section 2.1 looked at the current research into presentation of geospatial data, while sections 2.5 and 2.7 looked at the problem from the computer graphics, data visualisation and exploratory spatial data analysis (ESDA) angles. Anselin's work on ESDA [Ans99], which is discussed there, goes beyond the simple GIS and map visualisation of Maguire [Mag91], tending towards spatial analytics toolkits for the desktop like PYSAL [Rey+13]. Where Anselin talks about, "visualising local patterns of spatial association", this has not been achieved through analysis of a single map as he intends, but through analysis of thousands of maps through map comparisons (section 5.2 on data stores). All of this is achieved through the use of a web-based framework which is gradually extending into the CyberGIS infrastructure favoured by Wang [Wan10]. Architecture aside, though, the most difficult part of this to achieve is the web-based mapping and spatial analytics, mainly due to limitations of browser technology.

To put this into context, significant advances have been made in the automatic map building services, which function to a high degree without any user intervention. This technology enables the data store mining and map comparisons on which a large part of this thesis depends. This is a new concept, which does not appear in the literature review framed in this way. The map comparisons of Muller [Mul75] are all between pairs of maps, with no hint that large-scale spatial correlation to produced a graph of relationships is considered.

The problem of analysing real-time data was originally framed as 'real-time GIS', consisting of contiguous frames of spatial data. Once the problem is framed in an agent-based modelling context, with the rules formed from how agent states are altered between frames as the data under analysis, then the nature of the problem is changed. Automatic map generation and automatic model generation are not too dissimilar. Rand's, "when not to go into a bar" example [Ran06] is just one paper on learning rules from a real-time stream, but it characterises a distinct lack of literature on this subject. Agent-based modelling is unusual in this area, where modellers would normally propose the rules themselves and use a model for simulation and testing. The other example of learning in ABMs referenced in the literature review is the 'doppelganger' approach of Hoog [Hoo17] using deep neural networks. This differs from Rand's example, being more abstract in concept. Apart from this, the main literature on this subject appears to be in the domain of economic modelling.

9.1.1 Q1.1 Web GIS

Q1.1 What computer architecture satisfies the "WebGIS" requirement, taking into account storage and performance?

The basic premise behind the research in section 5.1, "Data Exploration and Web GIS", is that increases in computing power, coupled with the relaxation of certain restrictions on the dissemination of vector data, have made handling raw geospatial data on the web possible. While current systems have focused on cartography and tangible data, the opportunity is there to expand MapTube with more advanced spatial analytics functionality. In its initial form, two maps could only be compared visually, which was achieved by laying them on top of one another and fading the colours in and out. The term used in the JISC Techwatch publication [Bat+10a] is "Map Mashup", but the "MapTubeV" vector tiler project in section 5.1.4, and earlier work on the Mood Maps, demonstrates the viability of the new web GIS approach.

The literature review sections on “Linking GIS and Computer Graphics” (section 2.7) and “Computer Graphics and Rendering” (section 2.5) both cover the client side of the data presentation in the browser. Although the choropleth form of map is arguably the most popular on the web, other forms of visualisation exist, as highlighted in the literature review. Section 5.1.3, “Exploratory Data Analysis: Weather Data”, demonstrated that bespoke visualisations are viable. The work linked conventional web-based map rendering with a client side javascript WebGL rendering library, “three.js”, with the result of displaying weather data symbols and coverage data containing radar pictures on Google Maps. This is the same technological infrastructure as the MapBox GL JS systems uses, although with MapBox, the rendering capability is limited to more typical geographic data. This was explored in section 5.1, “Data Exploration and Web GIS”.

The theory behind web-based mapping is introduced in section 4.2, “Tiled Maps”, which is followed by a description of the Mood Map concept, starting with the first Radio Four map about the “Credit Crunch”. Directly following this, in section 4.4, “Dynamic Maps”, the server architecture is expanded into a ‘render on demand’ system capable of handling millions of hits an hour, proving that the theory presented here works in a practical system. This forms the technological basis on which the automatic map generation and data store mining questions depend.

While there can be no optimum architecture, the storage and performance of this system has been tested to the extreme in a working application, proving the speed and resilience of the design that was borne out of the theoretical performance in chapter 4. While this holds for the server side, on the client side, the pace of development in browser technology has meant that the application side has constantly had to change and evolve, as this is where the biggest gains can be made through thick client architectures. This is a good thing for spatial analytics on the web in general, as the increased interactivity with data in the browser now means that all the pieces are there for web-based spatial analytics. It just needs current web-based mapping systems to catch up with the latest technological leap.

9.1.2 Q1.2 Map Comparisons

Q1.2 The ability to build maps automatically from data and populate a geospatial data store raises questions about how connections and similarities between maps can be formed, leading to the discovery of new knowledge about

the structure of these relationships.

The point made earlier in this chapter was that the original map comparisons of Muller [Mul75] in "Associations in Choropleth Map Comparison", were intended to be between pairs of maps, not large-scale spatial cross correlation. The task here was to show that the time required for what is an $\mathcal{O}(n^2)$ problem could be reduced through performance gains and different methods of comparison.

Firstly, before comparison can take place, a bulk upload of suitably related data is required. By automating the identification of geospatial data, all 2,558 variables from the 2011 Census were added to MapTube as new maps and potential new data sources. Comparing this to the earlier project[1] to map 145 variables from the 2001 Census in 2009, this marks a step change in the ability to handle data. The main point made by figure 9.2 is that automatic map generation has the potential to create data at a scale that makes automatic analysis an essential tool for researchers.

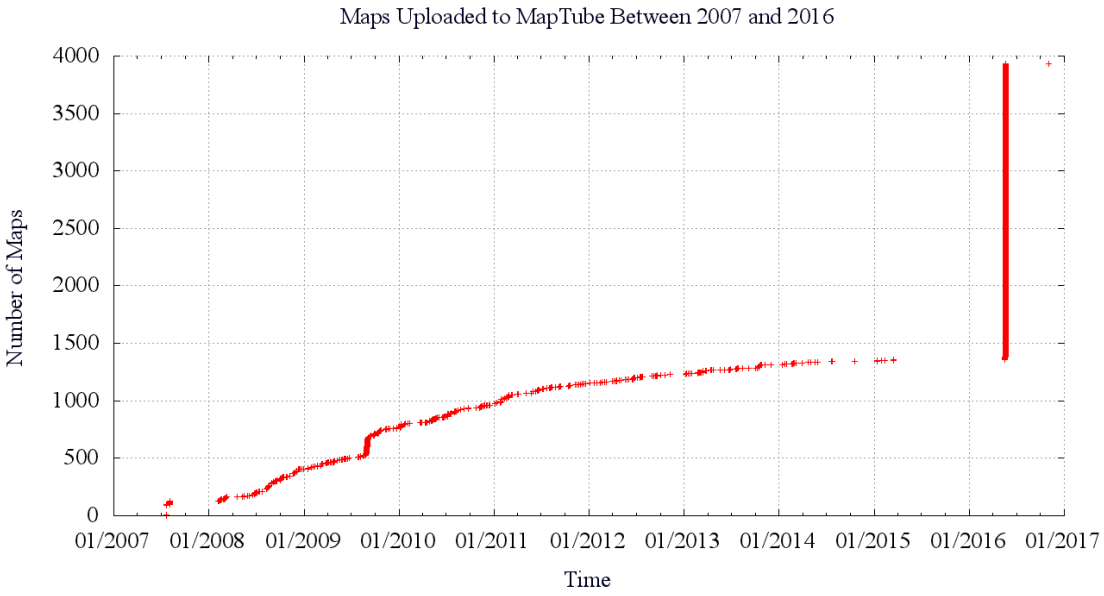


Figure 9.2: All maps uploaded to MapTube between 2007 and 2016.

The automatic identification relies on a store of boundary data along with detection of every column type using data mining techniques. This can be computationally expensive on large datasets, but geospatial data of this nature is bounded by the geographic aggregation level. The finest level Census data contains 181,000 areas for England and Wales, which is within the capabilities of the system. Having implemented

[1] Funded as part of the JISC National e-Infrastructure for Social Science (NeISS) Census Mapping Project, see: *http://www.maptube.org/about.aspx*.

this system for choropleth data based on $(areakey, value)$ tuples, it has revealed a potential next step in expanding the system to handle alternative representations of geospatial data. For example, the origin and destination flow maps from the travel to work data mentioned in section 8.5 on unified data, or data in the form of tables and Excel spreadsheets are all potential candidates.

An unexpected result from using the 2011 Census as the test case was caused by the large number of related variables. This is, in essence, what is stated in the abstract about the "analysis of a connected fabric of data". The data store mining was originally expected to be centred around outlier detection, but it proved harder to detect potential candidates in amongst the high number of *obvious* positive correlations, which is why the keyword analysis was added in Chapter 5. The development of this part of the thesis followed the sequence of making it easier for people to upload maps to MapTube with automatic map creation (Chapter 4, "Designing Systems"), then expansion into data store mining, correlation and visualisation of the results (Chapter 5, "Dynamic Visualisation"), leaving Chapter 7, "Data Exploration: Data Stores and Correlation", to apply all the techniques to the Census data, different data geographies and other stores of data.

In bringing in text domain analysis to the correlation methodology, another dimension is added to the results. This work uses the older "bag of words" approach, rather than Mikolov's word vectors [Mik+13]. The reason for this is that the text correlations performed on the Census data here pre-dates Mikolov's word vectors paper by a year. The "Knowledge Discovery" section of the literature review (section 2.10) also references other deep neural network methods for natural language processing, for example the sequence to sequence work of Sutskever [SVL14] on machine translation, which have value here if the correlation experiment is ever repeated. However, the Census example is restricted by its lack of high quality metadata for all variables, instead using a table based approach to describing the data, which severely limits the text domain correlation. This is not the case with other maps on MapTube, though.

Following on from the literature review in section 2.8, "Comparing Map Data and Correlations", different approaches to the correlation problem were covered, resulting in a comparison between the computationally efficient k-nearest neighbour algorithm (section 7.1.2) and spatial cross correlation, which is presented in the results section 7.1, "Data Stores and Correlation". A prediction of computation speed based on number of

zones and number of maps is included here. The analysis follows the graph theory reviewed in section 2.10 on "Knowledge Discovery", covering graphs and linked data.

In terms of how the static data on MapTube relates to other geospatial data, though, the tools presented in section 5.2, "Example 2, Data Stores", are potentially very effective in dealing with the example data. The problem with spatial cross correlations with large numbers of maps is the combinatorial explosion and time taken for this to run on the computer. This is where the "Pattern Matching" in section 5.2.2 and the k-nearest neighbour algorithm in section 7.1.2 were used as a heuristic to reduce the number of comparisons. The most interesting result from this research is shown in the graph of figure 7.7 (reprinted here as figure 9.3), which shows the correlation coefficients of the first three variables in the Census on a graph against each other. The graph is similar to a parallel coordinates plot, showing the similarity of the three Census variables by how similar they are to each other. This is potentially an important discovery if it provides a "fingerprint" of a dataset based on how it relates to existing data.

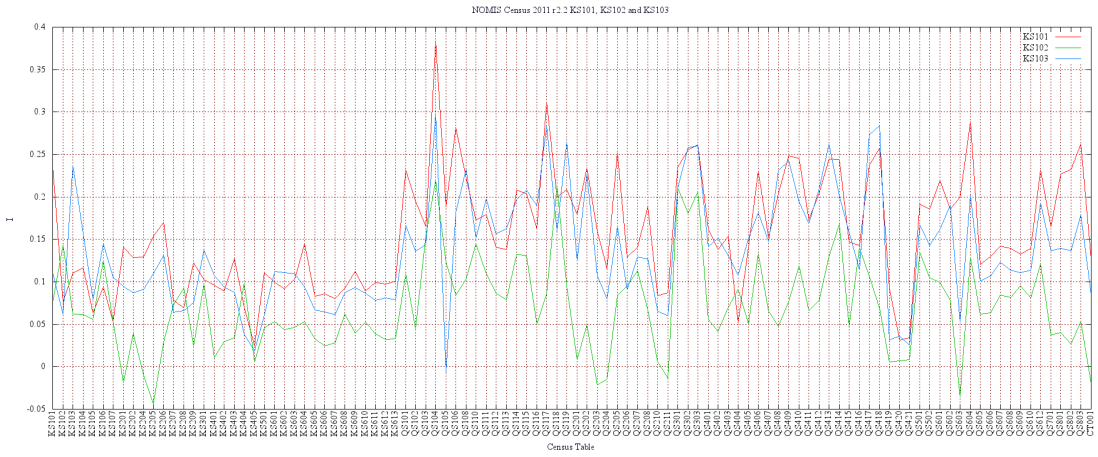


Figure 9.3: Graph of correlation coefficient (I) for first three tables (KS101-KS103) against all tables. KS101 is the 'Usual Resident Population' table, KS102 is 'Age Structure' and KS103 is 'Marital Status'.

Finally, taking into consideration the correlation results above, one potential improvement to the problem of computational efficiency in map comparisons is to match against a set of indicator patterns first. Section 5.2.2 on "Pattern Matching" used a self-organised neural network to build a set of 16 template maps using the Census 2011 data. This avoids the problem of the time required to add a new map increasing linearly as the number of maps increases, but also introduces another potential question. If all the data relates to the same place, then an outlier map that relates to nothing else might be an indicator of a missing link in the data? In other words, is it possible to know that

an essential element is missing from the relationships between maps? The idea of what we know, "known knowns" and "known unknowns", was never fully explored due to time constraints.

9.1.3 Q1.3 Real-time GIS and Agent-based Models

> *Q1.3 Real-time geospatial data is a geospatial agent-based model evolving over time. Can the rules that make the system work be learnt from the real-time stream of data?*

The work on "real-time mapping" and the development of the "Advanced Networks for complex Transport Systems (ANTS) project" referred to in the abstract is contained in Chapter 6, "Real-time Mapping and Agent Simulations". While originally intended to centre around the emerging real-time sources of data about London, the potential to combine an agent based modelling approach with data mining introduced a new element to the analysis. The diagram in figure 6.4, adapted from the paper by Rand on "Machine Learning Meets Agent Based Modelling" [Ran06], was a factor in this decision. Having amassed several years' worth of transport and environmental data about London, the lack of tools with which to analyse it suggested that a new approach was needed. In the literature review section on "Agent-based Modelling for Real-time Data" (section 2.4), systems like "FlameGPU" and Improbable's "SpatialOS" were referenced, but these are only simulation systems, deferring the analysis to other techniques which must act on the data they produce. Wilensky and Rand's paper on replication in models [WR07] comes closer, providing a set of best practices for models which actively encourages comparisons between models, but here again, the details of how to analyse the rules of the models are missing.

Both Chapter 6 and Chapter 8, "Data Exploration: Real-time and Data That Moves", rely heavily on building up an analysis using all the data collected in 2014. Trains, tubes and buses can be identified as agents in an ABM, but the logic behind how the model works can either be hard-coded by the programmer, or learnt from the real-time data stream. When building the first tube models, it was discovered that much of the information about position, direction and route choice was consistently wrong, which led to thinking about the minimum amount of a priori information required. Then, moving on to modelling the bus network, the bus stop locations and routes in the official TfL dataset contain multiple errors (see figure 8.3, "Two pictures of a complex transport network"). Learning the behaviours from the real-time stream is introduced in section

6.3, "Behaviours and Agent Based Models", which then continues to show how route choice and agent creation and destruction can be extracted from the 2014 archive. The interesting result here is that the pure network graph of the connections between the nodes (stations) on the network can be learnt from the stream, but their actual geographic locations cannot[2]. While this chapter is an interesting first step in real-time GIS, the work needs to progress further in order to be generally useful. A problem here is the lack of standards and protocols covering this type of data. Going back to the tube, bus and rail networks, geographically positioned nodes and network graphs of origin destination links need to have well-defined standards. One method, which exists now, is to use a shapefile containing lines between nodes. The approach used in Chapter 6 is to handle the pure network graph separately, but link to a table of geographically positioned stations through a unique station code key. The Open Geospatial Consortium's "Moving Features Specification" [Ope15] is one method for specifying agents which move, but not agents which are travelling along a network. The archive of temporal data is also a problem, but the simple strategy of using folders structured as "year/month/day" with data contained in files named according to observation time was employed. This is simple enough to be universally applicable to a wide variety of uses.

The "Learning from Real-time Streams" in section 2.9 of the literature review covered a wide variety of methods, with the examples generally falling into the area of reinforcement learning e.g. [WWT99] and [Hoo17]. The literature also stressed the necessity of validation in models. The doppelganger which van der Hoog discusses in his "Deep Learning in (and of) Agent-Based Models: A Prospectus" was proposed in relation to economic models. Validation of models where the model is a 'predictor/corrector' model of a physical system of moving agents is shown to be a less complicated problem. The tube network model in chapter 6 performs a comparison between the real world and model world (following [Ran06]) every three minutes. Therefore, a metric is available which shows how much correction has been necessary and how well the model is following the real world. This is also available spatially, but more work is needed in this area of real-time analysis techniques.

[2]Learning the physical location of a tube station when only the time taken to travel between adjacent stations is known might be possible with partial information about nearby stations and triangulation techniques. This remains an interesting research question, "could the relative geographic positions of tube stations be discovered using machine learning techniques if the only information available was the amount of time a tube took to travel between stations and the origin destination links making up the network graph?".

To conclude, the real-time GIS chapters make a start at creating tools for real-time data analysis, but this research needs to be taken further. Minimising the a priori information required though learning the behaviour from the data stream is a potentially powerful approach and complements the data mining and feature detection in the data exploration chapter which is used for exploration and analysis. The main result is in section 8.4, where figures 8.19, 8.20 and 8.21 show the NO_2 levels for London averaged over the whole of 2014. The morning and evening peaks link with the rush hour peaks plotted from the real-time transport data archive and the gate entry and exit peaks from the archived TfL dataset (figure 8.13). The chapter finishes by exploring the programmable element of the ABM approach, where a virtual sensor of bus density around an air quality sensor is created. The theoretical background for this comes from the discussion of using bus probe data in [Uno+09]. This idea is only mentioned briefly, but forms an important link between modelled and interpolated data and sensor data (figure 8.28). Following on from this, the analysis is repeated with data for days when bus strikes took place to determine their effect on air quality (figures 8.31 and 8.32). It is this ABM approach which allows the investigation of these linked city systems through the combination of the real-time sources of data.

9.1.4 Q1.4 Combined Sources of Data

Q1.4 What links exist between static and real-time data analysis?

The tag line on the MapTube website has always been, "a place to put maps", which is a reference to its aim of crowd-sourcing geospatial data. The "data store mining" covered earlier is one approach to the analysis of the structure of "the overall fabric of data" from the abstract, but the combination of static and real-time data is another. This was first referred to in section 6.4, where building height, street layout, wind direction and air quality were linked together. Without going to the complexity of a fluid dynamics model, wind direction and street layout affect air quality, together with traffic flow data. In the literature review, section 2.11, "Relative Scales in Data Comparison", the problem of interpolating data across weather fronts is mentioned in relation to comparisons between data sources [ACM16]. Transport, weather and air quality form a linked system, which was one of the aims of the data exploration in Chapter 8. After rush hour commute patterns in tube and bus data were analysed, the "Environment" section (8.4) demonstrates how these sources of data form a connected system. More research is needed before this is fully achieved, despite the fact that the

tools are now there to bring the separate sources of data together. The data on weather and air quality is not of sufficient temporal resolution and there is arguably a missing element of systems analysis theory which is required to show the connections. Both these reasons put the identification of systems in the data beyond the scope of this thesis, although it could still be argued that the link made between the rush hour peak commuter flow and the NO_2 hourly averages is one such link. This was only achieved by averaging over an entire year and removing the temporal resolution problem from the NO_2 data.

The original concept behind linking static with real-time data was the observation that the agent-based models in Chapter 6 require static data as inputs in order to function. This is also supported in the literature review of other models in section 2.9 on learning from real-time streams where a number of economic models are discussed. In section 2.4 on combining static and real-time data, the examples take a different approach, where Douglass used mobile phone data to derive population estimates which were then validated using the Census [Dou+15] and Smith-Clarke built poverty maps, again using mobile phone data [SMC14]. As was stated earlier, the key is the availability of data. Mobile phone calls provide connected data at scale and have been used extensively in publications, which the literature review picked up on. Real-time systems like the tube and bus systems also contain connected data about time and space. In order to build the models, though, data on the physical layout of track or road, stopping points, speed and timetables are needed. This is where the static and real-time are linked, as the flow of vehicles has been designed to follow expected commuter flow. Working population and employment should link to people's ability to get to work, which is where section 8.5, "Unified Data" finished off the real-time data analysis chapter. The tube entry and exit figures plotted in figure 8.13 are another example, as peak commuter flows from static data, taken in the context of real-time tube vehicle data, give passenger loading estimates. Using the spatial context to the real-time data, it is possible to determine where these problems are occurring. This is the type of analysis that is carried out regularly by the organisations who run the transport networks, and so is not new. Where this work stands out is in the derivation of the parameters of the model from the real-time stream (static data), which can then be analysed. As an example, take tube creation and destruction (figure 6.10), which gives service frequency, which then combines with travel to work statistics from the Census. The real-time and

static comparison comes from the data that is used to inform the model, which has been learnt from the stream. Also, this model calibration data may well change over time, which is a metric of the system that can also be used for analysis.

In conclusion, much of the analysis relating to this question is limited by the availability of data. The question itself is possibly too restricted, as, in addition to comparing static with real-time, static to static and real-time to real-time also need to be considered. While the original idea was that real-time models needed static data to run, and real-time models produced static data that could be analysed, the question has expanded to the point where the lines between the two are blurred.

9.2 Key Contributions

The following list contains the key technical and theoretical contributions of this work:

1. A web-based infrastructure for automatic mapping. The work here also fed into another web-based project called QUANT: *https://quant.casa.ucl.ac.uk*, which would not have been possible otherwise.

2. A new concept of data store mining.

3. Map comparisons on a large-scale to discover relationships in spatial data.

4. Preliminary work on learning models from real-time streams.

9.3 Work Presented in this Thesis

Linking back to the "Software" Section in the preface, the following is a list of all the software presented in this thesis, along with my contribution. Where there was any collaboration with other researchers, this is noted.

Project	Contribution
GMapCreator	Code, concept, design, implementation
MapTube	Code mine, graphic design by an artist
MapTubeD	Code, concept, design, implementation
GeometryFinder	Code, concept, design, implementation
MapTubeV	Code, concept, design, implementation
Data Store Miner	Code, concept, design, implementation
GeoGL	Code, concept, design, implementation
MapTubeExplorer	Code, concept, design, implementation
Meteorological Observation Decoder	Code, concept, design, implementation
Adaptive Networks for complex Transport Systems (ANTS)	Code and concept mine, project a collaboration between three others
CityDB API	Code, concept, design, implementation, with the project feeding into Ollie O'Brien's *http://citydashboard.org* project and Steven Gray's iPad wall project, plus some MSc projects

Table 9.1: Contribution of Work Recorded in this Thesis

9.4 Limitations

The web architecture presented in this thesis addresses the problem of browser-based geospatial data presentation, along with adding functionality to make it easier for the general public to make their own maps from data and so provide a service to crowdsource maps. With 3,446 maps and 16,898 users, this system has undergone a significant amount of testing in the real world. One problem with the automatic map generation and user created content is that the users might not be experts in geospatial data. Firstly, MapTube can only draw choropleth maps, heatmaps, points and lines.[3] This is a restriction that can be overcome without any technical difficulties on the server side where the tiles are rendered, but, on the client side and the web site, it requires work on a new user interface. While this is not difficult, it requires time to develop, and there has never been a requirement from users to include other visualisations. Cartograms would be a challenge, though, due to the way they warp the map to display the data.

The second limitation with automatic mapping and inexperienced users concerns the presentation of the data. Maps on MapTube are published immediately, without

[3]This is not strictly true, as shapefiles can be uploaded containing custom geometry created from a GIS, but MapTube is principally designed to handle areas with attributes.

moderation, so it is perfectly possible for users to plot maps that skew the data, whether intentionally or unintentionally. The area units might not be the best chosen for the data, but that can not be controlled as the data is presented to the system 'as is'. If a user presents it with a ward level map when lower layer super output (LSOA) area works more effectively, then it is impossible for MapTube to force the data to the finer scale. This is the modifiable areal unit problem in [Ope84]. However, the user can see how the map looks on the preview screen and the automatic system will detect continuous or discrete breaks, pick the colours and show a cumulative density plot of the data. One further improvement could be to also show the goodness of fit of the data classes, which are initially natural breaks by default, but can be changed to uniform and quantiles by the user. The number of breaks also defaults to either five for continuous data, or the number of detected classes for discrete data. This links back to Calka's work on comparing choropleth map classes [Cal18]. The idea of "long-tail breaks" is not covered by MapTube, apart from allowing the user to throw away the automatic breaks and create them manually. Having said this, though, QGIS does not have this type of feature either. In designing a system of this nature, there are two routes that can be followed. Either the system can function as an 'expert' and lock the user in to following a path chosen by the designer that guarantees[4] a well designed map, or the automatic mapping system can be seen as providing a first guess for the user, who then has complete control to do anything, whether good or bad. MapTube follows the second path, allowing the user total flexibility in all the parameters, so it is the responsibility of the user to display the data to best effect. As MapTube is a crowd-sourcing system, it is perfectly possible for another user to pick up a map, change how it is displayed and re-publish it. Sometimes the data can be flawed, too, as in the BBC Look East Survey on anti-social behaviour where the categories were "great community and no problems", along with "no problems", "boy racers" and "noisy neighbours".

On the subject of crowd-sourcing maps, one of the problems discovered early on is that users are less than ideal when it comes to describing what their maps show. In terms of map metadata, MapTube requires a title, keywords, short description and long description. The spatial extents it can work out itself, along with some properties of the attribute data. The system forces something to be entered, but cannot detect whether it is a description or just random text. Where a user enters 'TBA', or 'to be added', and

[4]Guaranteeing a well designed map is impossible unless it can be proved mathematically that the designer has accounted for every possibility. 100% completeness is unlikely to be realistic.

then does not add what the map shows, then the data is effectively useless. Possible methods to overcome this limitation are expanded on in the following section.

Having dealt with the automatic generation and presentation of maps, the architecture behind how the tile rendering works needs to be considered. MapTube has always been designed to be an Internet data mapping system. In section 4.1.2 on "Internet Maps", this distinction was made between MapTube and Google's Fusion Tables, which was released afterwards. On Fusion Tables, the data must be uploaded to Google's server first, but with MapTube, the web services work using URI endpoints for data. This means that MapTube downloads its data directly from the Internet on demand, avoiding the need to have a large local store, but requiring a complex buffering and caching system instead. This gives MapTube greater flexibility with Internet based data, for example data stores, but at a cost. One unforeseen problem occurred with a map for a charity website that was created in CASA and built maps from crowd-sourced data. Something was wrong with the code on the computer collecting the data, which proceeded to 'spam' MapTube's tile renderers with a URI for data that never fulfilled the request. The net result of this is that the central control code of the MapTubeD tile renderer entered a locked state, waiting for the http request to either be satisfied, or time out after 30 seconds. The bug that had not been anticipated, though, was that this would lock out all tile rendering for all maps, effectively performing a denial of service on the map rendering system. While a total redesign of the thread locking mechanism behind MapTubeD's control system solved the issue, the problem remains that loading data from the Internet takes time, incurring a load penalty every time a new map, not in the most recently used cache, is requested. With small size attribute data files, which the system was originally designed around, this is not a problem, but large files containing custom geometry suit the upload architecture rather than the load on demand one. MapTube therefore allows upload of large files, which can then be rendered using the render on demand system from a local route, which is faster than from the external Internet. With the MapTubeV system for vector tiles, there is an additional problem involving the initial reprojection of the data when it is first loaded. The original solution was to write a fast reprojection algorithm, but with large data sets, reprojection can take minutes, which is unacceptable. A test was tried with the OS data for all the building outlines in the south east tile covering England (i.e. London down to Brighton, containing millions of buildings) to ascertain the level of data that could be handled.

While the rendering handles this level of data without problems, the loading time is prohibitive. An obvious next step with this software is to incorporate a reprojection on demand, rather than trying to reproject everything on load.

With the web-based mapping architecture that underpins MapTube, automatic generation of maps by mining data stores marks an important next step in geospatial analysis. This needs to be expanded as the current system is limited in the types of input data that it can automatically map from. Shapefiles account for structured data, while CSV files in table format account for the 'unstructured' data. MapTube expects one, or zero, header lines, followed by rows of data representing areas. Each column is an attribute associated to that area. One unexpected problem that came to light was that many spatial data sets contain an index column. This was the driving factor behind including a variety of column statistics to prevent the automatic mapping system from trying to make maps from uninteresting data. It is this system for identifying and labelling columns that could be expanded in line with the research in [WPH04] on understanding table structure. A wider range of input formats could then be accommodated.

Following on from automatic generation, the data store mining concept takes the work in an interesting direction, but is not a finished product in any sense. The *data.gov.uk* data store from the U.K. Government was used, but turned out not to have much in the way of spatially referenced data. Most data was in the form of textual tables and reports, although the mining of the metadata turned up some interesting information and anomalies. Although links to data advertised as open data were on the site, the link was often to another agency's website with the data behind an authentication system. The London Datastore *data.london.gov.uk* provided easy access to open data that was mappable, and so this was used as one of the first tests. After this, the NOMIS Census 2011 release was used for all subsequent work, treating the bulk upload on the site as if it was a data store. There remains the question of how many data stores there are in the world where this technique could be applied. The World Bank (*data.worldbank.org*) was considered, along with the U.S. Census, which is accessible as an RDF triple store. Google's Fusion tables and ESRI's ArcGIS online both contain large amounts of geospatial data, but licensing and access restrictions prevent these being data mined in this way. This is the main reason that the project is still at the test stage as an application on GitHub. It needs a set of data stores with related data to form a compelling use case. However, the expansion into handling bulk data as if it were a data store opens up a

wider variety of possibilities.

The concept of making large-scale comparisons between maps was the subject of chapter 7, “Data Exploration: Data Stores and Correlation”. The literature review in section 2.8, “Comparing Map Data and Correlations” picked up on the references which showed that two types of map comparison are possible: a visual human comparison between the look of two maps [LS77] and a mathematical correlation based on a function applied to each map’s attributes [Mul75]. While chapter 2.8 only considers the mathematical map comparisons, this is by design. The visual comparison of Lloyd [LS77] is designed to optimise the presentation of maps for a human audience, and, while his approach to modelling noise is an interesting method, mathematical correlations are required.[5] The remainder of the literature review deals with the computational aspects of making large scale map comparisons, but the type of comparison being performed could have been explored further. This was due to time constraints and the fact that this is not a system for performing a single type of map comparison, but a framework to allow researchers to test their own. The science of map comparisons could be the subject of a PhD topic in its own right, so the requirement was to use a well known correlation algorithm for testing, while attempting to speed up the computational time. In terms of speeding up the comparisons, code improvements were made to the point where the estimate of 87 days from section 5.2, “Example 2: Data Stores”, could be run inside 24 hours. This came from loading the data in blocks and parallelising the main loop, while at the same time investigating algorithmic improvements in the form of k-nearest neighbour testing. The issue of how far across space to attempt to cross correlate two datasets needs further research, but this is a data problem rather than an application problem. This is exactly the type of experiment that the framework was designed to allow other researchers to investigate. Finally, the idea of ‘pattern matching’ data was used to speed up the matching. The literature review mentions the problem of correlation not being transitive, which led to the idea of pattern matching against a small number of template maps. More research is required in this area, but the map comparison chapter achieved its desired aims, to prove that map comparison is both viable and useful as a technique.

[5] On a related note, the unit tests for the MapTubeD tile rendering code actually contain algorithms which transform the images into a mathematical image space designed to mimic human vision so that Euclidean distance between pixel values in this space corresponds to how similar a human vision system would perceive them to be. The reason that this was required was because it was discovered that the test tile images used in the unit tests could be rendered slightly differently on different systems, so the map tiles being used as the gold standard test of correct rendering were not always an exact match. While RGB space and Euclidean distance could have been used, a vision space called L_{ab} was felt to be a better test (see: [RP11]).

Chapter 6, “Real-time Mapping and Agent Simulations”, provided real-time data for London. This facet of the thesis provided the ability to handle data that was too big for the web-based mapping systems to handle. In this respect, Chapter 8, “Data Exploration: Real-time and Data That Moves”, shows that this was successful with the quantity of data required for analysis. The original idea was to use an agent-based modelling system, allowing researchers to query parameters of the system while it runs. This proved to be more complicated than anticipated, due to errors in the real-time stream and missing data. Much of chapter 6 goes towards solving the problem of which line a tube train is running on, as this information is not included in the stream. In fact, the data from TfL is designed around users who want to know the next train arriving at the station where they are currently waiting, not an overview map of the entire network. This is a common problem and also manifests itself in the bus data and bike hire data, while weather data and rail data provide APIs for querying everything in a single operation. A significant portion of the work went into circumventing these problems rather than analysing the data. Also, there is a hidden problem discovered during the development of the London Underground network model. Where the model relies on the published speeds of the tubes on the network using the timetables, if the model is then used to calculate average speed, this will be invalid. It depends on how the model is written and what information is being used. As an example, the real-time data contains the expected time of arrival of a tube, but this is not necessarily the time it arrives if it travels slowly. After arriving at the station, before the next data update, the model has to have it leave before another expected arrival at the next station is available and so uses the average speed for that link until the real data is available. This produces a bias in the calculation of average line speeds from the real-time data, but, by using a separate model designed only to use archive data, this is made possible. Essentially, the archive only model can peek into the future to see how long a tube actually took to traverse a link and so produce accurate data. Admittedly, this is also possible in the real-time model, but only with additional code to manage the information and dependencies. In the literature review of section 2.2, “Agent-based Modelling for Real-time Data”, this problem never appeared. If linking models together is ever to be a reality, then more research is needed on exactly this type of issue. While the simulation possible using an agent-based methodology is very powerful, the implementation can hide the details in a way that affects the result, which is alluded to by Wilensky and Rand’s set

of best practices for agent-based modelling, “Making Models Match: Replicating an Agent-Based Model” [WR07]. For the time being, though, the technique is best used with models that are small and simple enough to allow inspection of the algorithms. The idea of being able to ‘publish’ model algorithms that other researchers can use is also limited by this problem, but documentation of how models work would go a long way towards mitigating the potential issues. Finally, the application of one transport model, for example, the London Underground, to another transport model, for example the London Bus Network, was another idea that came from the conclusion of chapter 6. In one sense a model of this type might be seen as universal, if the data inputs and outputs are identical. All the tube model does is to move vehicles around a graph network following updates from the real-time stream. If the movement code were seen as universal, then comparing the operation of different networks can be built around this framework. More research is needed in this area.

9.5 Further Work

Where the automatic generation of maps from data is concerned, there is a wide scope for expanding the work into other formats of data. The system described here is restrictive in the way it expects files containing data formatted as rows and columns. The work of Wang, in “Table structure understanding and its performance evaluation” [WPH04], shows that this can be expanded to detection of mappable data contained in documents. With a more intelligent data processing work flow, a natural extension is to allow the system to search the whole Internet for data, rather than just selected data stores. Linked to this are the recent advances in natural language processing, for example Mikolov in “Distributed Representations of Words and Phrases and their Compositionality” [Mik+13] and the long short term memory (LSTM) models, gated recurrent units (GRUs) and sequence to sequence models which are currently employed in text recognition and translation [HS97] [SVL14]. This increased ability to handle natural language is also a potential step forward in the comparison of maps where the correlations are made on the descriptions of what the maps contain.

A related problem of missing map descriptions was also mentioned in the limitations. The general public is not always completely thorough in describing the data behind maps, but, in making large scale map comparisons, there might be ways of detecting what a map shows without any describing data. This is more of an academic

question, but it asks whether it is possible to detect what data is contained on the map in the absence of any metadata. As it stands now, it can be described on the basis of how closely it follows existing, known data. For example, is it a population effect or is it related to jobs and employment, or maybe it is an entirely new feature?

With the crowd-sourcing of maps on MapTube, the presentation of the maps highlights the problems identified in the literature review on colours and classification breaks. Here, an idea to expand the system and improve the automatic generation abilities is to make MapTube able to learn from users. When a user uploads a map, the automatic mapping system shows an initial preview of how it looks. The user can then override the colours and breaks to improve the visualisation. If MapTube was able to learn from how its initial guess was being improved, then it would lead to the system slowly improving over time. Other forms of visualisation apart from choropleths, point and heatmaps could also be included.

Returning to the subject of map correlations, this can be related to the problem of completeness and unknowns. MapTube has a "Topicality Index" built into it which is used to promote maps to the front page if they contain data about stories currently in the news[6]. By using this in reverse, it is possible to find topics in the news that MapTube does not have data for and either flag it for the user to find, or build a system which can automatically search for data. In essence, the system could be made to look for gaps in its own data by looking at connections between the data that it owns and topics of interest in the outside world.

The idea behind building a real-time GIS system also deserves further exploration. As sources of real-time data grow in number, so there will be more use cases around which to build a general purpose system. At the moment, the real-time GIS presented here is more of a closely coupled agent-based model, data mining tool and visualisation system. The 3D visualisation system is built around the "programmable maps" idea where the developer has the flexibility to write code that interacts and interrogates the model as it is running. This, coupled with a visualisation layer, allows experiments to be run on live data. For example, "show me all the late running trains", or "show me NO_2 levels interpolated using a dispersion model and correlated with road traffic density". Identifying a set of building blocks of this nature, which could be connected together into a work flow, could then be labelled as a true real-time GIS system. The previous

[6]MapTube's "Topicality Index" is built using keywords from news stories in the BBC and Guardian RSS feeds in much the same way as the text mining work on the Census in section 5.2.1, "Correlation in the Text Domain".

section mentioned the idea of universal models, or models of systems which can be applied in different contexts. For example, the tube model moves vehicles around a network graph, which is exactly the same as the bus model, only with a different graph. More research is needed on this agent-based modelling approach to real-time systems, but the limiting factor has turned out to be a lack of real-time data feeds. The aim of this part of the research is to be able to bring expert knowledge into the system in the form of the programmed models, while providing the data which researchers need to run their own experiments.

Finally, looking at the overall picture of the work presented in this thesis, the work required to put these ideas into practice needs to be taken into account. The web-based mapping changes are mainly implementation changes, requiring new map visualisation modules to be developed and deployed. The natural language processing is a more complex problem, but is self-contained as specific improvements in well-defined areas, so is something that could be achievable in the short term. The expanded data processing workflow is more of a research project, requiring a new methodology, while expansion of the cross correlation could potentially be implemented on the current website as the computational problems appear to be solvable. Adding moving agents has been a plan for some time, as this will enable real-time data to be displayed on the web. Security problems exist with allowing user uploaded models to be run, but the intention with developing the C++ code as an application was always to apply the lessons learned to the website.

Chapter 10

Conclusion

The introduction in Chapter 1 began with the technological advances which have made geospatial simulation and visualisation unrecognisable from a generation ago. As with the ensemble forecasting in Lewis, [Lew05], which had to wait until computing had advanced sufficiently in the 1990s before it was viable, data can now be analysed and models run on a scale which was not previously possible. While the games industry may have been responsible for pushing the development of Graphics Processing Units (GPUs) for visualisation, which then became general purpose processing units, the increased use of smart phones and apps gives rise to a new wave of human generated data. The relaxing of some of the licensing restrictions on vector data in the UK has precipitated a switch towards handling geospatial data in its raw format, allowing an unprecedented level of interaction and exploration. The future is difficult to predict given the current pace of technological progress, but the next generation processing systems might exploit new advances in hardware designed for Deep Learning and AI, with the ability to perform complex real-time analytics. While this might be a possible future, this research is about what geocomputing systems can do now.

After introducing GIS, real-time data and Internet data stores, the background in Chapter 3 delved into the evolution of computing architectures, benchmarks and geospatial computing in more depth. Despite DARPANET being conceived in 1969, the modern Internet age has only been around since the mid 1990s and is consequently still in its early twenties. The 'cloud' and CyberGIS are even younger, so the geospatial techniques of Fisher and Jenks from the 1960s and 70s are still evolving on modern computer hardware. The 'HyperLogLog' and other frugal algorithms from the cloud computing and stream mining world have yet to fully migrate into large scale geospatial computing algorithms. When Fisher stated in his 1958 paper [Fis58] that partitioning

methods based on frequency distribution are more suitable for large datasets, he only had limited computing power at his disposal. Modern computing often makes problems like this trivial through sheer 'instructions per second', when techniques which achieve a 95% correct answer in 50% of the time might be more suitable. The network graph methods and spatial cross correlation highlighted in this chapter are both $\mathcal{O}(n^2)$ problems, based on an $n \times n$ matrix of every possible link or dataset comparison. The data store correlation and similarity examples show the problems of finding links between thousands of maps (n), which quickly turns into millions of similarity tests (n^2). While parallel computation might speed this up by a factor of 100 or 1000, depending on the number of processing units, this is an outlier detection case with only a small percentage of positive matches. The ability to experiment on the existing data to discover which heuristics work in speeding up the discovery of potential matches is what the Data Store Miner project has delivered. It is the ability to combine different sources of geospatial data together which is new, leveraging the rise in Internet data stores to provide the source of data to experiment on.

The literature review in Chapter 2 and technical background in Chapter 3 covered existing GIS systems from Symap to CyberGIS, web mapping and geospatial standards, finishing up with visualisation and data mining. This formed the basis for the next chapter, 'Designing Systems: Algorithms and Work Flows', where the mechanics of making maps from intangible and real-time data were introduced. Automatic map generation from data contained in *CSV* files was first mentioned here, in the context of making it easier to collect geospatial data and publish it with the MapTube website. With the realisation that data mining techniques can be applied to an unknown dataset to identify its spatial context and map the data automatically, the idea behind the 'Data Store Miner' was conceived. The other key idea here is that web-based mapping corresponds to the visualisation layer in a conventional desktop GIS. The heterogeneous web services used to power systems like MapTube and SurveyMapper can be re-purposed into a more general GIS framework. Rather than being simple 'data presentation' systems, they can be turned into 'data exploration' systems with complex analytical functions. This is where the identification of fundamental algorithms in geospatial computing becomes important, along with existing standards for inter-operability. Real-time data was also mentioned at this point, with APIs able to provide data on what is happening in a city on a minute by minute basis. The combination of this type of real-time data

with the more established static datasets like the 2011 Census is a new research topic.

Having established a clear direction for research, Chapter 4, 'Designing Systems: Algorithms and Work Flows', moved on to designing a cloud-based, or 'CyberGIS', system from the existing tile rendering architecture of the MapTube website. Beginning with the MapTube website which was originally conceived as a presentation system, work flows for volunteered geographic information which began as the Radio Four 'Credit Crunch Mood Map', and which turned into the Survey Mapper website, are the genesis of a more complex geo-processing architecture. These systems were proven in their ability to handle millions of hits per hour, handling map drawing requests in a load-balancing and edge cached environment which could be adapted to build a more generally useful 'web GIS' system. Weather data was used as a real-time example of standards based web GIS, along with the introduction of the 'ANTS' project which collects data about real-time transport systems in London. Together, these systems form the basis of the data source on which the rest of the chapters depend. Add to this the work flow for automatic map creation and the Data Store Miner and the groundwork is in place for the following chapters to capitalise on this diverse and extensive source of information. By using data mining techniques to map data automatically, it follows that more data will be unlocked from large Internet data dumps. An additional bonus was the realisation that automated map making also leads to more powerful searching of data stores. By using the automatic map making process to correlate the data in both the description and data domains, links can be made between data that might be related. While this is only 'correlation and not causation', the data mining processes used to make the maps can be termed 'search analytics'. The chapter ended by taking this work to its natural conclusion and the fact that the static data from MapTube and the dynamic data from the ANTS project provide the data sources for a real-time GIS system capable of showing what is happening in London in real-time. This was one of the aims stated in the original abstract, to create a GIS system which goes beyond a traditional desktop GIS and is capable of handling real-time data while allowing the user to experiment with it.

Now with some basic tools, Chapter 5, 'Dynamic Visualisation', filled in some of the gaps with the development of a Javascript library for overlaying animated data on top of Google Maps using 'ThreeJS', or any of the other Javascript games libraries built around WebGL. In addition to this, weather data was used as an example with radar

and synoptic data demonstrating a web GIS system for coverage data (radar, satellite or temperature colour fill) and bespoke visualisation for the wind fleches used in World Meteorological Organisation 'Form 12' surface synoptic plots. The rendering of bespoke graphics on maps using WebGL is an important feature to demonstrate as it takes a different architectural route to other tiled map systems. Also related to this was the library designed to couple the AgentScript agent-based modelling library with Google Maps, resulting in a geographic ABM running on a web page which is able to show real-time tube positions. These examples all belong to the 'programmable maps' class, with the live tubes demonstration shown running as a new type of MapTube map. This marks the beginnings of a next step in web-based mapping with NetLogo type models running on web pages using real data from APIs in a similar way to how the 'Modelling4All' website[1] did with Java Applets wrapping NetLogo models which could be shared and experimented with. While this has almost been achieved here, more progress still needs to be made before it is possible to experiment on ABM models running from real data in this way. The limits here are imposed by the web browser and the fact that the AgentScript library is based on 'Canvas' and 'SVG' rather than the more powerful 'WebGL', although the authors of the library have started to address this fact. Experiments with browser based visualisations of tube, bus and train together showed that the scale of the data is a challenge and that a desktop application would be better suited to the more complex visualisation and experimentation. Finishing up the web mapping section was the technique of 'Vector Tiling'. This is an alternative to the technique used earlier for the synoptic weather observations and serves tiles containing the vector data covered by the tile extents rather than an image like a *jpeg* or *png* as in the original web-based maps. While the first system rendered the data onto a continuous canvas, the vector tiler is designed to work with planetary scale data using hierarchical level of detail (HLOD). This highlights the point made earlier in the introduction of Chapter 1 about the difference between tangible and intangible, cartography and symbology. Complex data of limited area is generally handled more elegantly with the first system, which is more like the rendering layer of a conventional desktop GIS, drawing on a single screen viewport, while the vector tiler is more suited to complex data that needs to be simplified and reduced. An example of a classic use case for a vector tiler is MSOA level data for England and Wales, for example, any of the 2011

[1] Modelling4All website: *http://m.modelling4all.org*.

Census maps, with the added benefit that tiles can be load balanced across a server farm for increased speed and scalability. Both approaches to geographic rendering are covered here, plus animated data on maps, which is a technique not widely used.

After developing a number of new web mapping tools, Chapter 5 finished up by taking the automatic map generation a stage further with graph construction from both the data domain and the meta-data domain. This resulted in a tool for analysing relationships in the data that are otherwise hidden in amongst the noise of a large number of trivially correlated variables. Learning was explored briefly with the Census data, using a neural network classifier to partition the 2558 variables into one of 16 classes. This was motivated by the question of scalability and how to significantly reduce the amount of time taken to find where a new map fits within an existing framework of linked maps. Taken in the context of the MapTube website, an interesting idea is whether a data mining process could be run overnight which attempts to find previously undiscovered relationships between data. It is easy to envisage a weekly report of potential findings which could then be investigated further for causal relationships. In the context of making it easier to bring data into the public domain, automatically link it to make searching easier and data mine it overnight to find relationships, this is an important result which turns Internet data dumps into potential new sources of knowledge. This chapter created the tools needed to perform this type of analysis, setting the scene for the later chapter on data exploration to fully capitalise on it. The chapter concluded with a system diagram which showed how all the pieces of technology link together into a final system.

After exploring web GIS and data stores, Chapter 6, 'Real-time Mapping and Agent Simulations', picked up where the previous chapters left off on agent-based models and simulation. Recognising the limits of web-based systems, this chapter builds a desktop 3D GIS application capable of running models based on real data from the ANTS collection system. The chapter started out by examining the London Underground system in detail and using some of the data mining and machine learning techniques mentioned previously to fill in the gaps in the published data regarding the layout of the stations forming the network and the route codes and tube lines. By building up origin/destination data from the real-time stream, the system could then learn the layout of the tube network and all the routes. This type of analytics is shown to be important later on when the data is used to investigate indicators of normal running or failures. Statistics

from the five years of data collected so far are used, with the 2014 data analysed in most detail. This allows a normal operating point for the tube network to be derived, with the GIS system able to highlight anything unusual happening in the data in real-time. The output here is a 3D desktop GIS which uses real-time data from the Internet as its input and can run models with an instrumentation layer enabling visual analytics functionality.

The two chapters on 'Data Exploration' put all of the tools together to see what they can be used to achieve. In 'Data Exploration: Data Stores and Correlation', starting with the Census 2011 dataset, an in-depth analysis of all the static Census data is made. This is followed by a similar example examining all the maps on the MapTube website. This goes a stage further by using the automatic map creation tools to upload all 2558 mapped variables before using a data extraction service created for the MapTube site to extract all the data for comparison. This is a comparison of data with different base geometries, which is a technological step beyond what was accomplished with the Census in the earlier chapter. The key tool here is the ability to construct a similarity metric, whether k-nearest neighbour, spatial cross correlation or Bayesian comparison, and use the MapTube website as an environment in which to run the experiment. Rather than embedding the code into MapTube, the data is extracted via a web service, in keeping with the idea of heterogeneous geospatial services which can be chained together into an experimentation work flow.

The second data exploration chapter, 'Data Exploration: Real-time and Data That Moves', deals with real-time data, with models created for the London Underground, London bus network, National Rail trains, weather and air quality. The interactions between agents in these separate models are unknown and this is what the modelling and instrumentation is intended to discover. In most forms of scientific experimentation, the design of the experiment is such that only one variable changes in response to the environment while being measured. In a system as complex as a city, this is not possible, but by collecting data on a regular basis, the effect of tube strikes and bus strikes can be investigated. It is also possible to set space constant and see how weather and air quality vary over time, or set time constant and see how they vary over space. This is when the static data stored on the MapTube site can be brought in to look at transport data in relation to population density or travel to work statistics. Looking at the air quality data, the main result is how the NO_2 levels aggregated over an entire

year correlate with the transport data. The morning and evening rush hour peaks are an expected feature of any transport data. The software enabled these rush hour peaks to be identified in the real-time streams of tube, bus, rail and air quality data, linking together the different sources of data.

10.1 The Future

Looking forward, it is impossible to say where this research might lead in the next 5 or 10 years, given the current pace of change. Companies like D-Wave[2], IBM[3] and Google[4] are all doing research into practical Quantum computing architectures, so, in 10 years time, running millions of iterations of a spatial cross correlation might be as fast as a single iteration? While current parallel architectures like GPGPU and cloud computing using MapReduce are effective, certain classes of problem, like large scale map cross correlations, require a fundamental shift in the architecture. The target audience is the general public for the more basic visualisation, while the computationally expensive spatial analytics are targeted at providing infrastructure to support academic researchers. This is one area where the proliferation of cloud services has brought applications which were originally in the desktop domain to a wider audience via the Internet. The idea that an Internet site can store city data and models which researchers can use to experiment with is a very powerful one. It applies the 95% + 5% rule, where the researcher only needs to supply the 5% of data essential to their experiment and the rest comes from the web site. While the majority of data on MapTube is currently U.K. based, it has always been envisaged as a global mapping website, and this is certainly demonstrated by the users and maps on the site. In recent years, research around whether a model is universal enough to apply to a city in a different country and comparisons between different models has been enabled through increasingly open and accessible data. For example, in "Variability in Regularity: Mining Temporal Mobility Patterns in London, Singapore and Beijing Using Smart-Card Data." [C+16], the authors analyse smart card data for London, Beijing and Singapore. Two of the main barriers to worldwide research are the language barrier and local knowledge about how to access the data. The language is an interesting problem that has not been mentioned up to this point. The MapTube website uses English as its main language, but some users have uploaded maps with descriptions in different languages. In this situation,

[2]D-Wave: *https://www.dwavesys.com/quantum-computing*.
[3]IBM Q: *https://www.research.ibm.com/ibm-q/*.
[4]Google Quantum AI: *https://ai.google/research/teams/applied-science/quantum-ai/*.

if the map looks to be useful to others, then the description has been translated using Google Translate and appended onto the end of the native description. Maps are currently published immediately with moderation only occurring after the event. The scale of map uploads and type of data has not yet warranted an automated moderation or warning system, but language detection and automatic translation is a feature that has been considered. Automation is the key to handling data and research at scale. The techniques presented in this thesis go some way in this direction, but this is only the beginning of the creation of tools which take an intellectual step away from direct data analysis, towards more advanced data exploration at a higher level of abstraction. To finish where this section began, the future is impossible to predict, but advanced spatial analysis can only get more interesting from here.

10.2 Final Thoughts

In conclusion, this thesis has covered a wide variety of techniques relating to geospatial data analysis in order to develop a collection of tools. The "Data Store Miner" and the "MapTube Explorer" projects are designed around experimenting and searching for relationships between significant sized sets of related data. How this data relates is part of the experimentation process. The "MapTube Vector Tiler" project is geared more towards visualisation of one-off exploratory analysis of data, taking web based mapping one step closer to a web GIS. The Internet is where data stores and real-time data reside, so these steps enable questions to be asked about what is happening in a city now. With the quantity of real-time data being generated, the approach taken is to highlight the unexpected and the unusual. The aim of this research was to create worlds of data which could be explored and discovered. The key to this is making the data and tools available to researchers, which is what the MapTube website does. At the moment, MapTube has had 16,898 registered users.[5] The test of whether this is the correct approach will be seen by whether the number of users of MapTube goes up or down.

[5] 16,898 MapTube users is correct as of 12 March 2017. A user has to register in order to upload a map.

Bibliography

[Aba+16] Martin Abadi et al. “TensorFlow: A system for large-scale machine learning”. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283. URL: *https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf*.

[ACM16] Juliana Antunes Azevedo, Lee Chapman, and Catherine L. Muller. “Quantifying the daytime and night-time urban heat island in Birmingham, UK: A comparison of satellite derived land surface temperature and high resolution air temperature observations”. In: *Remote Sensing* 8.2 (2016). ISSN: 2072-4292. DOI: *10.3390/rs8020153*. URL: *http://www.mdpi.com/2072-4292/8/2/153*.

[ALW05] S. Alfarano, T. Lux, and F. Wagner. “Estimation of Agent-Based Models: The Case of an Asymmetric Herding Model.” In: *Computational Economics* 26.1 (2005), pp. 19–49.

[AM95] I. Aleksander and H. Morton. *An Introduction to Neural Computing*. Second Edition. International Thomson Publishing, 1995.

[AMD11] AMD. *Aparapi*. Advanced Micro Devices. 2011. URL: *http://developer.amd.com/tools-and-sdks/heterogeneous-computing/aparapi/*.

[Ami02] Isaac Amidror. “Scattered Data Interpolation Methods for Electronic Imaging Systems: A Survey”. In: *Journal of Electronic Imaging* 11.2 (Apr. 2002), pp. 157–176. DOI: *10.1117/1.1455013*.

[Ans12] Luc Anselin. “From SpaceStat to CyberGIS, Twenty Years of Spatial Data Analysis Software”. In: *International Regional Science Review* 35.2 (Apr. 2012), pp. 131–157. DOI: *10.1177/0160017612438615*. URL: *http://irx.sagepub.com/content/35/2/131.abstract*.

[Ans95] L. Anselin. “Local Indicators of Spatial Association - LISA”. In: *Geographical Analysis* 27.2 (1995), pp. 93–115.

[Ans99] Luc Anselin. “Interactive techniques and exploratory spatial data analysis”. In: *Geographical Information Systems: principles, techniques, management and applications* 1 (1999), pp. 251–264.

[Apa14a] Apache Software Foundation. *Apache Storm*. 2014. URL: *http://storm.apache.org*.

[Apa14b] Apache Software Foundation. *MESOS*. Apache Software Foundation. 2014. URL: *http://mesos.apache.org/*.

[Apa16a] Apache Software Foundation. *Apache Hive*. 2016. URL: *https://hive.apache.com*.

[Apa16b] Apache Software Foundation. *Apache Pig*. Apache Software Foundation. 2016. URL: *https://pig.apache.org*.

[Apa16c] Apache Software Foundation. *Apache Spark*. 2016. URL: *http://spark.apache.org*.

[Art94] W. Brian Arthur. "Inductive Reasoning and Bounded Rationality". In: *The American Economic Review* 84.2 (1994), pp. 406–411. ISSN: 00028282. URL: *http://www.jstor.org/stable/2117868*.

[AZB18] Sofiane Abbar, Tahar Zanouda, and Javier Borge-Holthoefer. "Structural robustness and service reachability in urban settings". In: *Data Mining and Knowledge Discovery* 32.3 (May 2018), pp. 830–847. ISSN: 1573-756X. DOI: *10.1007/s10618-018-0551-4*. URL: *https://doi.org/10.1007/s10618-018-0551-4*.

[BA99] L. Barabasi and R. Albert. "Emergence of scaling in random networks". In: *Science* 286 (1999), pp. 509–512.

[Bar16] Albert-Laszlo Barabasi. *Network Science*. Cambridge University Press, July 2016, p. 475. ISBN: 978-1107076266. URL: *http://networksciencebook.com*.

[Bat+08] M. Batty et al. "Scaling and allometry in the building geometries of Greater London". In: *The European Physical Journal B* 63 (2008), pp. 303–314. DOI: *10.1140/epjb/e2008-00251-5*.

[Bat+09] M. Batty et al. "The Neogeography of virtual cities: digital mirrors into a recursive world". In: ed. by M. Foth. IGI Global Snippet, 2009.

[Bat+10a] M. Batty et al. *Data mash-ups and the future of mapping*. Tech. rep. JISC Techwatch, Sept. 2010. URL: *http://www.jisc.ac.uk/media/documents/techwatch/jisc_10_1.pdf*.

[Bat+10b] Michael Batty et al. "Map Mashups, Web 2.0 and the GIS Revolution". In: *Annals of GIS* 16.1 (Mar. 2010), pp. 1–13. DOI: *10.1080/19475681003700831*.

[Bat+13] M. Batty et al. "Imagining the Future City: London 2062". In: London: Ubiquity Press, 2013. Chap. Smart London, pp. 31–40. DOI: *http://dx.doi.org/10.5334/bag.d*.

[Bat+15] Michael Batty et al. "Innovations in Digital Research Methods". In: Sage, June 2015. Chap. 11, pp. 245–270.

[Bat18] Michael Batty. "Digital twins". In: *Environment and Planning B: Urban Analytics and City Science* 45.5 (2018), pp. 817–820. DOI: *10.1177/2399808318796416*. eprint: *https://doi.org/10.1177/2399808318796416*. URL: *https://doi.org/10.1177/2399808318796416*.

[Ben75] J. L. Bentley. "Multidimensional binary search trees used for associative searching". In: *Communications of the ACM* 18.9 (Sept. 1975), pp. 509–517.

[BH09] Luiz André Barroso and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool, 2009.

[BHH03] Cynthia A. Brewer, Geoffrey W. Hatchard, and Mark A. Harrower. "ColorBrewer in Print: A Catalog of Color Schemes for Maps". In: *Cartography and Geographic Information Science* 30.1 (2003), pp. 5–32. DOI: *10.1559/152304003100010929*.

[BHM07] H. P. Boswijk, C. H. Hommes, and S. Manzan. "Behavioral heterogeneity in stock prices". In: *Journal of Economic Dynamics & Control* 31.6 (2007), pp. 1938–1970.

[Bir+10] Mark Birkin et al. "Elements of a computational infrastructure for social simulation". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 368.1925 (2010), pp. 3797–3812. DOI: *10.1098/rsta.2010.0145*. URL: *http://rsta.royalsocietypublishing.org/content/368/1925/3797*.

[Bir+11] M. Birkin et al. "Calibration of a Spatial Simulation Model with Volunteered Geographic Information". In: *International Journal of Geographical Information Science* (2011).

[Blo70] Burton H. Bloom. "Space/Time Trade-offs in Hash Coding with Allowable Errors". In: *Commun. ACM* 13.7 (July 1970), pp. 422–426. DOI: *10.1145/362686.362692*. URL: *http://doi.acm.org/10.1145/362686.362692*.

[Bos16] Mike Bostock. *D3 Library*. 2016. URL: *https://d3js.org*.

[Boy91] "MJRTY - A Fast Majority Vote Algorithm". In: *Automated Reasoning: Essays in Honor of Woody Bledsoe, Automated Reasoning Series*. Ed. by R. S. Boyer. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1991, pp. 105–117. URL: *http://www.cs.utexas.edu/users/boyer/mjrty.ps.Z*.

[Bra07] Max Bramer. *Principles of Data Mining*. UTiCS. Springer, 2007.

[C+16] Zhong C. et al. "Variability in Regularity: Mining Temporal Mobility Patterns in London, Singapore and Beijing Using Smart-Card Data." In: *PLoS ONE* 11.2 (2016). DOI: *https://doi.org/10.1371/journal.pone.0149222*.

[Cal18] Beata Calk. "Comparing continuity and compactness of choropleth map classes". In: *Geodesy and Cartography* 67.1 (2018), pp. 21–34. DOI: *10.24425/118704*. URL: *https://doi.org/10.24425,118704*.

[CC08] CASA and CSAP. *GENerative E-SocIal Science*. Centre for Advanced Spatial Analysis. 2008. URL: *http://www.genesis.ucl.ac.uk*.

[CC18] Guido Caldarelli and Michele Catanzaro. *Networks, A very short introduction*. Oxford University Press, 2018, p. 122. ISBN: 978-0-19-958807-7.

[CGL79] Tony F. Chan, Gene H. Golub, and Randall J. LeVeque. *Updating Formulae and a Pairwise Algorithm for Computing Sample Variances*. Tech. rep. Stanford, CA, USA: Stanford University, 1979.

[Cha+06] Fay Chang et al. “BigTable: A Distributed Storage System for Structured Data”. In: *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*. OSDI ’06. Berkeley, CA, USA: USENIX Association, 2006, pp. 15–15. URL: *http://dl.acm.org/citation.cfm?id=1267308.1267323*.

[Che+13] James Cheshire et al. “CyberGIS: Fostering a New Wave of Geospatial Discovery and Innovation”. In: Springer-Verlag, 2013. Chap. CyberGIS for Analyzing Urban Data. URL: *http://eprints.ncrm.ac.uk/3159/*.

[Che+14] J. Cheshire et al. “CyberGIS: Fostering a New Wave of Discovery and Innovation.” In: Springer-Verlag, 2014. Chap. CyberGIS for Analyzing Urban Data.

[Cho15] François Chollet. *keras*. https://github.com/fchollet/keras. 2015.

[Chr04] Nick Chrisman. *Charting the Unknown. How Computer Mapping at Harvard Became GIS*. ESRI Press, 2004.

[Cim+19] Giulio Cimini et al. “The Statistical Physics of Real-World Networks”. In: *Nature Reviews Physics* (Jan. 2019). DOI: *10.1038/s42254-018-0002-6*. URL: *https://arxiv.org/abs/1810.05095*.

[CM04] Graham Cormode and S. Muthukrishnan. “An Improved Data Stream Summary: The Count-Min Sketch and its Applications”. In: *J. Algorithms* 55 (2004), pp. 29–38.

[CO70] Andrew D. Cliff and Keith Ord. “Spatial Autocorrelation: A Review of Existing and New Measures with Applications”. In: *Economic Geography* 46 (1970), pp. 269–292. ISSN: 00130095, 19448287. URL: *http://www.jstor.org/stable/143144*.

[Cod72] E. F. Codd. “Relational completeness of data base sublanguages”. In: *Database Systems*. Prentice-Hall, 1972, pp. 65–98.

[CR11] Patrick Cozzi and Kevin Ring. *3D Engine Design for Virtual Globes*. Taylor and Francis Group, 2011.

[Cre93] Noel A. C. Cressie. “Spatial Prediction and Kriging”. In: *Statistics for Spatial Data*. John Wiley and Sons, 1993. ISBN: 9781119115151. DOI: *https://doi.org/10.1002/9781119115151.ch3*.

[Cro+09] A. T. Crooks et al. “Crowdsourcing Spatial Surveys and Mapping”. In: *Proceedings of the 17th Geographical Information Systems Research UK Conference*. Ed. by D. Fairbairn. 2009.

[DG08] Jeffrey Dean and Sanjay Ghemawat. “MapReduce: simplified data processing on large clusters”. In: *Commun. ACM* 51.1 (Jan. 2008), pp. 107–113. ISSN: 0001-0782. DOI: *10.1145/1327452.1327492*. URL: *http://doi.acm.org/10.1145/1327452.1327492*.

[DGM08] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. “Critical phenomena in complex networks”. In: *Reviews of Modern Physics* 80 (2008), pp. 1275–1335.

[DM15] M. D'Lima and F. Medda. "A new measure of resilience: An application to the London Underground". In: *Transportation Research Part A - Policy and Practice* 81 (2015), pp. 35–46. ISSN: 0965-8564. DOI: *http://dx.doi.org/10.1016/j.tra.2015.05.017*. URL: *http://discovers.ucl.ac.uk/1472487*.

[Dor96] Danny Dorling. "Area Cartograms: Their Use and Creation". In: *Concepts and Techniques in Modern Geography* (1996).

[Dou+15] Rex W. Douglass et al. "High resolution population estimates from telecommunications data". In: *EPJ Data Science* 4.1 (May 2015), p. 4. ISSN: 2193-1127. DOI: *10.1140/epjds/s13688-015-0040-6*. URL: *https://doi.org/10.1140/epjds/s13688-015-0040-6*.

[EA96] J. M. Epstein and R. L. Axtell. *Growing Artificial Societies: Social Science From the Bottom Up*. Brookings Institutional Press, 1996.

[EJX01] D. W. Embley, D. Jackman, and L. Xu. "Multifaceted exploitation of metadata for attribute match discovery in information integration". In: *Workshop on information integration on the Web*. 2001. URL: *https://www.deg.byu.edu/papers/wiiw01.pdf*.

[EP09] J. M. Epstein and J. Parker. "Large Datasets - detail and overview problems of scale - planetary data". In: *Nature* (2009). URL: *http://www.nature.com/nature/journal/v460/n7256/full/460687a.html*.

[ER59] P. Erdos and A. Renyi. "On random graphs I." In: *Publicationes Mathematicae (Debrecen)* 6 (1959), pp. 17–61.

[ESR98] ESRI. *ESRI Shapefile Technical Description*. ESRI. July 1998. URL: *http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf*.

[ET94] Bradley Efron and R. J. Tibshirani. "An Introduction to the Bootstrap". In: (May 1994), p. 456.

[Eur14] European Petroleum Survey Group. 2014. URL: *http://spatialreference.org*.

[Eve05] Gerald I. Evenden. *libproj4: A Comprehensive Library of Cartographic Projection Functions*. Mar. 2005. URL: *http://trac.osgeo.org/proj/*.

[Fan+08] Wenbin Fang et al. "Parallel Data Mining on Graphics Processors". In: *Technical Report HKUST-CS08-07* (Oct. 2008). URL: *http://code.google.com/p/gpuminer/*.

[Far86] G. Farin. "Triangular Bernstein-Bezier Patches". In: *Computer Aided Geometric Design, Elsevier* (1986), pp. 83–127.

[Fie00] Roy Fielding. "Architecture Styles and the Design of Network-Based Software Architectures". PhD thesis. University of California, Irvine. Information and Computer Science., 2000.

[Fis58] Walter D. Fisher. "On Grouping for Maximum Homogeneity". In: *American Statistical Association Journal* 53 (1958), pp. 789–798.

[Fly66] M. J. Flynn. “Very high-speed computing systems”. In: *Proceedings of the IEEE* 54.12 (Dec. 1966), pp. 1901–9219. DOI: *10.1109/PROC.1966.5273*.

[Fol+93] J. Foley et al. *Computer Graphics, Principles and Practice*. Second Edition. Addison Wesley, 1993.

[Gam+99] Erich Gamma et al. *Design Patterns*. Addison Wesley, 1999.

[Gas12] Benedict R. Gaster. *Heterogeneous Computing with OpenCL*. 2nd ed. ISBN-13: 978-0124058941. Morgan Kaufman, 2012.

[GB34] C. E. Gehlke and H. Biehl. “Certain effects of grouping upon the size of the correlation coefficient in census tract material”. In: *Journal of the American Statistical Association, Supplement 29* (1934), pp. 169–170.

[Gea54] R. C. Geary. “The Contiguity Ratio and Statistical Mapping”. In: *The Incorporated Statistician* 5 (1954), pp. 115–45.

[Gib+08] M. Gibin et al. “Collaborative Mapping of London Using Google Maps: The London Profiler, Number 132”. In: *CASA Working Papers* (Mar. 2008), pp. 1–15. URL: *http://www.bartlett.ucl.ac.uk/casa/pdf/paper132.pdf*.

[Gib95] William Gibson. *Neuromancer*. Harper Collins, 1995, p. 67.

[GMH13] S. Gray, R. Milton, and A. Hudson-Smith. “Visualizing real-time data with an interactive iPad video wall”. In: *National Centre for Research Methods ePrints* Spring (2013), p. 7. URL: *http://eprints.ncrm.ac.uk/3066/1/MethodsNewsSpring2013.pdf*.

[GMH15] Steven Gray, Richard Milton, and Andrew Hudson-Smith. *Advances in Crowdsourcing*. Springer, 2015, pp. 163–179. DOI: *10.1007/978-3-319-18341-1*.

[GMS17] M. Gangappa, C. Kiran Mai, and P. Sammulal. “Techniques for Machine Learning based Spatial Data Analysis: Research Directions”. In: *International Journal of Computer Applications* 170.1 (July 2017), pp. 9–13. ISSN: 0975-8887. DOI: *10.5120/ijca2017914643*. URL: *http://www.ijcaonline.org/archives/volume170/number1/28032-2017914643*.

[Gon+10] Hector Gonzalez et al. “Google Fusion Tables: Data Management, Integration and Collaboration in the Cloud”. In: *Proceedings of the 1st ACM Symposium on Cloud Computing*. SoCC ’10. Indianapolis, Indiana, USA: ACM, 2010, pp. 175–180. ISBN: 978-1-4503-0036-0. DOI: *10.1145/1807128.1807158*. URL: *http://doi.acm.org/10.1145/1807128.1807158*.

[Gri16] Michael Grieves. “Origins of the Digital Twin Concept”. In: (Aug. 2016). DOI: *10.13140/RG.2.2.26367.61609*.

[Hal+09] Mark Hall et al. “The WEKA Data Mining Software: An Update”. In: *SIGKDD Explorations* 11.1 (July 2009), pp. 10–18.

[Hay94] S. Haykin. *Neural Networks, A Comprehensive Foundation*. New York: Macmillan College Publishing Company, 1994.

[Heb49] Donald O. Hebb. *The Organization of Behaviour: A Neuropsychological Theory*. New York: Wiley, 1949.

[Hen10] Benjamin Hennig. *General Election 2010: Swings that did matter*. School of Geography and the Environment, University of Oxford. May 2010. URL: *http://www.viewsoftheworld.net/?p=736*.

[Hep+12] Alison J. Heppenstall et al. *Agent-Based Models of Geographical Systems*. Springer Netherlands, 2012, p. 760. DOI: *10.1007/978-90-481-8927-4*.

[Hil15] Maarten Hilferink. *Fisher's Natural Breaks Classification*. PBL Netherlands Environmental Assessment Agency. 2015. URL: *http://wiki.objectvision.nl/index.php/Fisher%27s_Natural_Breaks_Classification*.

[Hoc05] H. Hochmair Hartwig. "Ontology Matching for Spatial Data Retrieval from Internet Portals". In: *GeoSpatial Semantics*. Ed. by M. Andrea Rodríguez et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 166–182. ISBN: 978-3-540-32283-2.

[Hol95] John H. Holland. *Hidden Order, How Adaptation Builds Complexity*. Helix Books, 1995.

[Hoo17] Sander van der Hoog. "Deep Learning in (and of) Agent-Based Models: A Prospectus". In: (June 2017). URL: *https://arxiv.org/abs/1706.06302v1*.

[HP11] John Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Fifth Edition. Morgan Kaufman, 2011.

[HS00] C. Holscher and G. Strube. "Web Search Behaviour of Internet Experts and Newbies". In: *9th International World Wide Web Conference*. Amsterdam, Holland, 2000.

[HS97] Sepp Hochreiter and Jurgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[Hud+07a] A. Hudson-Smith et al. "Public Domain GIS, Mapping and Imaging using Web-based Services, Number 120". In: *CASA Working Papers* (Sept. 2007), pp. 1–12. URL: *http://www.bartlett.ucl.ac.uk/casa/pdf/paper120.pdf*.

[Hud+07b] A. Hudson-Smith et al. "Virtual Cities: Digital Mirrors into a Recursive World". In: *CASA Working Papers* 125 (Dec. 2007), pp. 1–25. URL: *http://www.bartlett.ucl.ac.uk/casa/pdf/paper125.pdf*.

[Hud+08] A. Hudson-Smith et al. "Mapping for the Masses: Accessing Web 2.0 through Crowdsourcing, Number 143". In: *CASA Working Papers* (Aug. 2008), pp. 1–19. URL: *http://www.bartlett.ucl.ac.uk/casa/pdf/paper143.pdf*.

[Hud+09a] A. Hudson-Smith et al. "NeoGeography and Web 2.0: Concepts, Tools and Applications". In: *Journal of Location Based Services* 3.2 (June 2009), pp. 118–145. DOI: *10.1080/17489720902950366*.

[Hud+09b] Andrew Hudson-Smith et al. “Mapping for the Masses, Accessing Web 2.0 Through Crowdsourcing”. In: *Social Science Computer Review* 27.4 (2009), pp. 1–15. DOI: *10.1177/0894439309332299*.

[IBM07] IBM. *Many Eyes*. IBM. 2007. URL: *http://www.ibm.com/manyeyes*.

[IJ93] Emmanuel C. Ifeachor and Barrie W. Jervis. “Digital Signal Processing, A Practical Approach”. In: Addison Wesley, 1993. Chap. The decimation-in-time fast Fourier transform algorithm, pp. 65–76.

[Imp18] Improbable. *SpatialOS*. *https://improbable.io*. 2018.

[Inn10] Peter Innes. *Understand the Weather: Teach Yourself*. Hachette UK, 2010.

[Jen77] George F. Jenks. “Optimal Data Classification for Choropleth Maps”. In: *Issue 2 of Occasional paper, University of Kansas Dept. of Geography-Meteorology University of Kansas Dept. of Geography Occasional Paper* 2 (1977).

[Jia+14] Yangqing Jia et al. “Caffe: Convolutional Architecture for Fast Feature Embedding”. In: *Proceedings of the 22Nd ACM International Conference on Multimedia*. MM ’14. Orlando, Florida, USA: ACM, 2014, pp. 675–678. ISBN: 978-1-4503-3063-3. DOI: *10.1145/2647868.2654889*. URL: *http://doi.acm.org/10.1145/2647868.2654889*.

[JW97] Karen Sparck Jones and Peter Willet. *Readings in Information Retrieval*. ISBN 1-55860-454-4. Morgan Kaufman, 1997. URL: *http://tartarus.org/martin/PorterStemmer/*.

[Keh+11] Michael Kehoe et al. *Smarter Cities Series: A Foundation for Understanding IBM Smarter Cities*. 2011. URL: *http://www.redbooks.ibm.com/redpapers/pdfs/redp4733.pdf*.

[Kel08] G. Kelly. “An Interactive System for Procedural City Generation”. MA thesis. Institute of Technology, Blanchardstown, 2008.

[Ker09] Nathan Kerr. “Alternative Approaches to Parallel GIS Processing”. MA thesis. Arizona State University, 2009. URL: *http://www.nathankerr.com/projects/parallel-gis-processing/alternative_approaches_to_parallel_gis_processing.pdf*.

[Khr14] Khronos Group. *WebGL Specification*. Khronos Group, Oct. 2014. URL: *https://www.khronos.org/registry/webgl/specs/1.0/*.

[Kir11] Alan Kirman. “Learning in Agent-based Models”. In: *Eastern Economic Journal* 37.1 (2011), pp. 20–27. ISSN: 00945056, 19394632. URL: *http://www.jstor.org/stable/41239490*.

[Kit14] Rob Kitchen. “The Real-time city? Big Data and smart urbanism”. In: *GeoJournal* 79.1 (2014), pp. 1–14. DOI: *doi:10.1007/s10708-013-9516-8*.

[Kli] A. Klinger. “Patterns and Search Statistics”. In: *Optimizing Methods in Statistics* (). Ed. by J. S. Rustagi, pp. 303–337.

[KM06] G. Kelly and H. McCabe. “A Survey of Procedural Techniques for City Generation”. In: *Institute of Technology Blanchardstown Journal* 14 (2006), pp. 87–130.

[KM16] Rob Kitchen and Gavin McArdle. “What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets”. In: *Big Data & Society* 3.1 (2016), pp. 1–10. DOI: *10.1177/2053951716631130*. eprint: *https://doi.org/10.1177/2053951716631130*. URL: *https://doi.org/10.1177/2053951716631130*.

[Knu00a] Donald E. Knuth. *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*. Vol. 2. Addison Wesley, 2000.

[Knu00b] Donald E. Knuth. *The Art of Computer Programming, Volume 3, Sorting and Searching*. Vol. 3. Addison Wesley, 2000.

[Knu96] Donald E. Knuth. *Selected Papers on Computer Science*. CSLI lecture notes 59. CSLI Publications, 1996.

[Kri51] D. G. Krige. “A statistical approach to some mine valuation and allied problems on the Witwatersrand”. In: *J. Chem. Metal, Min. Soc. South Africa* 52 (1951), pp. 119–139.

[KV01] Alan P. Kirman and Nicolaas J. Vriend. “Evolving market structure: An ACE model of price dispersion and loyalty”. In: *Journal of Economic Dynamics and Control* 25.3 (2001). Agent-based Computational Economics (ACE), pp. 459–502. ISSN: 0165-1889. DOI: *https://doi.org/10.1016/S0165-1889(00)00033-6*. URL: *http://www.sciencedirect.com/science/article/pii/S0165188900000336*.

[LB14] R. Louf and M. Barthelemy. “A typology of street patterns”. In: *Journal of the Royal Society, Interface* 11.101 (2014). DOI: *10.1098/rsif.2014.0924*.

[Lee44] L. P. Lee. “The Nomenclature and Classification of Map Projections”. In: *Empire Survey Review* 7.51 (Jan. 1944), pp. 190–200.

[Lew05] John M. Lewis. “Roots of Ensemble Forecasting”. In: *American Meteorological Society* 133 (2005), pp. 1865–1885.

[LFB17] Jing Li, Michael P. Finn, and Marta Blanco Castano. “A Lightweight CUDA-Based Parallel Map Reprojection Method for Raster Dataset of Continental to Global Extent”. In: *ISPRS International Journal of Geo-Information* 6.4 (2017). ISSN: 2220-9964. DOI: *10.3390/ijgi6040092*. URL: *http://www.mdpi.com/2220-9964/6/4/92*.

[Lou10] Mike Loukides. *What is Data Science?* O’Reilly Strata. ISBN: 978-1-449-32755-2. Mar. 2010.

[LRW15] Shaun Larcom, Ferdinand Rauch, and Tim Willems. “The Benefits of Forced Experimentation: Striking Evidence from the London Underground Network”. In: *University of Oxford Department of Economics* 755 (Sept. 2015). ISSN: 1471-0498. URL: *https://www.economics.ox.ac.uk/materials/papers/14046/paper-755.pdf*.

[LS77] Robert Lloyd and Theodore Steinke. “Visual and Statistical Comparison of Choropleth Maps”. In: *Annals of the Association of American Geographers* 67.3 (1977), pp. 429–436. ISSN: 00045608, 14678306. URL: *http://www.jstor.org/stable/2562340*.

[LT16] Henry W. Lin and Max Tegmark. “Why does deep and cheap learning work so well?” In: *arXiv* (2016). DOI: *arXiv:1608.08225*.

[Lyn91] P. A. Lynn. *An Introduction to the Analysis and Processing of Signals*. Third Edition. Macmillan, 1991.

[Mag91] D. J. Maguire. “An overview and definition of GIS”. In: *Geographical information systems Vol. 1: principles* (Jan. 1991), pp. 9–20.

[McC68] Richard B. McCammon. “The Dendrograph: A New Tool for Correlation”. In: *Geological Society of America Bulletin* 79.11 (1968), pp. 1663–1670. DOI: *10.1130/0016-7606(1968)79[1663:TDANTF]2.0.CO;2*.

[MCL10] David Martin, Samantha Cockings, and Samuel Leung. “24-hour gridded population models”. In: *European Forum for Geostatistics Conference* (Oct. 2010).

[MG09] James R. Miller and Tom Gaskins. “Computations on an Ellipsoid for GIS”. In: *Computer-Aided Design and Applications* 6.4 (2009), pp. 575–583. DOI: *10.3772/cadaps.2009.575-583*. URL: *http://people.eecs.ku.edu/~miller/Papers/CAD_6_4__575-583.pdf*.

[MGH13] R.. Milton, S. Gray, and A. Hudson-Smith. “CyberGIS for Analyzing Urban Data”. In: *National Centre for Research Methods ePrints* Winter (2013), p. 3. URL: *http://eprints.ncrm.ac.uk/3217/1/MethodsNewsWinter2013.pdf*.

[Mik+13] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: *CoRR* abs/1310.4546 (2013). arXiv: *1310.4546*. URL: *http://arxiv.org/abs/1310.4546*.

[Mil+] Richard Milton et al. “Talking to Gnomes: Exploring Privacy and Trust Around Internet of Things Devices in a Public Space”. In: *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI EA ’18. Montreal, Canada: ACM, LBW632:1–LBW632:6. ISBN: 978-1-4503-5621-3. DOI: *10.1145/3170427.3188481*. URL: *http://doi.acm.org/10.1145/3170427.3188481*.

[Mil+18] Richard Milton et al. “Smart IoT and Soft AI”. In: *Living in the Internet of Things: Cybersecurity of the IoT - 2018*. Vol. CP740. London, England: IET Conference Publications, June 2018.

[Mil09] Richard Milton. “The Origins of MapTube”. In: *Civil Engineering Surveyor* (June 2009), pp. 30–32. URL: *http://www.civilengineeringsurveyor.com*.

[Mil95] G. A. Miller. “WordNet: A Lexical Database for English”. In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[MM05] L. Mitas and H. Mitasova. “Spatial Data Interpolation”. In: *Geographic Information Systems: Principles, Techniques, Management and Applications*. Ed. by P. A. Longley et al. 2nd Edition. Vol. 1.2. 2005. Chap. 34. URL: *http://www.geos.ed.ac.uk/~gisteac/gis_book_abridged/*.

[Mor50] P. A. P. Moran. “Notes on Continuous Stochastic Phenomena”. In: *Biometrika* 37.1/2 (1950), pp. 17–23. ISSN: 00063444. URL: *http://www.jstor.org/stable/2332142*.

[MS05] R. Milton and A. Steed. “Correcting GPS Readings from a Tracked Mobile Sensor”. In: *Location and Context Awareness, Lecture Notes in Computer Science* 3479 (2005), pp. 83–94.

[MS07] R. Milton and A. Steed. “Mapping Carbon Monoxide Using GPS Tracked Sensors”. In: *Environmental Monitoring and Assessment* 124.1-3 (2007), pp. 1–19.

[Mul75] Jean-Claude Muller. “Associations in Choropleth Map Comparison”. In: *Annals of the Association of American Geographers* 65.3 (Sept. 1975), pp. 403–413.

[MW15] Nathan Marz and James Warren. *Big Data: Principles and best practices of scalable real-time data systems*. Manning Publications, Apr. 2015, p. 328. ISBN: 9781617290343.

[Net15] Network Rail. *Data feeds - Network Rail*. Network Rail. 2015. URL: *http://www.networkrail.co.uk/data-feeds/*.

[Nie+16] M. Niebetaner et al. “Real-Time Rendering Techniques with Hardware Tessellation”. In: *Comput. Graph. Forum* 35.1 (Feb. 2016), pp. 113–137. ISSN: 0167-7055. DOI: *10.1111/cgf.12714*. URL: *https://doi.org/10.1111/cgf.12714*.

[Ope07] Open Geospatial Consortium. *Web Processing Service (WPS)*. Open Geospatial Consortium, June 2007. URL: *http://www.opengeospatial.org/standards/wps*.

[Ope10a] Open Geospatial Consortium. *OpenGIS Web Map Tile Service Implementation Standard*. Open Geospatial Consortium, 2010. URL: *http://www.opengeospatial.org/standards/wmts*.

[Ope10b] Open Geospatial Consortium. *Simple Feature Access - Part 2: SQL Option*. Open Geospatial Consortium, 2010. URL: *http://www.opengeospatial.org/standards/sfs*.

[Ope11] Open Geospatial Consortium. *GepSPARQL - A geographic query language for RDF data*. Open Geospatial Consortium, Jan. 2011. URL: *http://www.opengeospatial.org/standards/geosparql*.

[Ope15] Open Geospatial Consortium. *Moving Features*. Open Geospatial Consortium, Feb. 2015. URL: *http://www.opengeospatial.org/standards/movingfeatures*.

[Ope16] Open Source Geospatial Foundation. *Geographic Resources Analysis Support System*. OSGeo. Mar. 2016. URL: *https://grass.osgeo.org*.

[Ope84] S. Openshaw. “The Modifiable Areal Unit Problem”. In: *Concepts and Techniques in Modern Geography* 38 (1984), pp. 1–40. ISSN: 0306-6142.

[Ope97] Stan Openshaw. *Artificial Intelligence in Geography*. John Wiley and Sons, 1997, pp. 1–1. ISBN: 9780471969914. URL: *https://www.wiley.com/en-us/Artificial+Intelligence+in+Geography-p-9780471969914*.

[Ord10] Ordnance Survey. *A Guide to Coordinate Systems in Great Britain*. Dec. 2010.

[OSG13] OSGeo. *Geotools Library*. 2013. URL: *http://www.geotools.org*.

[Pas+17] Adam Paszke et al. "Automatic differentiation in PyTorch". In: *NIPS-W*. 2017.

[PE12] J. Parker and J. M. Epstein. "A Distributed Platform for Global-Scale Agent-Based Models of Disease Transmission". In: *Transactions on Modeling and Computer Simulation (TOMACS), Association for Computing Machinery (ACM)* 22 (2012).

[Per85] Ken Perlin. "An Image Synthesizer". In: *SIGGRAPH Comput. Graph.* 19.3 (July 1985), pp. 287–296. ISSN: 0097-8930. DOI: *10.1145/325165.325247*. URL: *http://doi.acm.org/10.1145/325165.325247*.

[PH90] David A. Patterson and John L. Hennessy. *Computer Architecture: A Quantative Approach*. Morgan Kaufman, 1990.

[Pha+05] Doantam Phan et al. "Flow Map Layout". In: *Proceedings of the 2005 IEEE Symposium on Information Visualization*. INFOVIS '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 29–. DOI: *10.1109/INFOVIS.2005.13*. URL: *http://dx.doi.org/10.1109/INFOVIS.2005.13*.

[Pir+02] P. Pirolli et al. "A user-tracing architecture for modeling interaction with the world wide web". In: *Advanced Visual Interfaces*. Trento, Italy, 2002.

[Por80] Martin F Porter. "An algorithm for suffix stripping". In: *Program* 14.3 (1980), pp. 130–137.

[Por91] Michael E. Porter. "America's Green Strategy". In: *Scientific American* 264.4 (Apr. 1991).

[Ran06] WILLIAM Rand. "Machine learning meets agent-based modeling: When not to go to a bar". In: *the Proceedings of Agent*. 2006.

[Rat+10] Carlo Ratti et al. "Redrawing the Map of Great Britain from a Network of Human Interactions". In: *PLOS ONE* 5.12 (Dec. 2010), pp. 1–6. DOI: *10.1371/journal.pone.0014248*. URL: *https://doi.org/10.1371/journal.pone.0014248*.

[Rea+16] J. Reades et al. "Finding Pearls in London's Oysters". In: *Built Environment Special Issue: Big Data and the City* 42.3 (Oct. 2016), pp. 365–381.

[Rei06] Michael Reilly. "Where to Find the Freshest Air in Town". In: *New Scientist* (Sept. 2006), pp. 26–27.

[Rey+13] Sergio J. Rey et al. "Parallel Optimal Choropleth Map Classification in PySAL". In: *International Journal of Geographical Information Science* 27.5 (2013), pp. 1023–1039. DOI: *http://dx.doi.org/10.1080/13658816.2012.752094*.

[Ric14] Paul Richmond. "Resolving Conflicts between Multiple Competing Agents in Parallel Simulations". In: *Euro-Par 2014: Parallel Processing Workshops*. Ed. by Luís Lopes et al. Cham: Springer International Publishing, 2014, pp. 383–394. ISBN: 978-3-319-14325-5.

[Rij+13] Jan N. van Rijn et al. “Machine Learning and Knowledge Discovery in Databases”. In: vol. 8190. Springer Berlin Heidelberg, 2013. Chap. OpenML: A Collaborative Science Platform, pp. 645–649. DOI: *10.1007/978-3-642-40994-3_46*. URL: *http://link.springer.com/chapter/10.1007%2F978-3-642-40994-3_46*.

[Riv+18] Gonzalo F Rivas-Torres et al. “A methodology for mapping native invasive vegetation coverage in archipelagos: An example from the Galapagos Islands”. In: 42.1 (2018), pp. 83–111. DOI: *10.1177/0309133317752278*. eprint: *https://doi.org/10.1177/0309133317752278*. URL: *https://doi.org/10.1177/0309133317752278*.

[Rou13] Flora Roumpani. “Developing classical and contemporary models in ESRI’s City Engine, Number 191”. In: *CASA Working Papers* (May 2013), pp. 1–19.

[RP11] Erik Reinhard and Tania Pouli. “Colour Spaces for Colour Transfer”. In: *Computational Color Imaging*. Ed. by Raimondo Schettini, Shoji Tominaga, and Alain Trémeau. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–15. ISBN: 978-3-642-20404-3.

[Sam06] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. 500 Sansome Street, Suite 400, San Francisco, CA 94111: Morgan Kaufman, 2006. ISBN: 978-0-12-369446-1.

[SD08] Charles R. Schmidt and Boris Dev. “A Scalable Tile Map Service for Distributing Dynamic Choropleth Maps”. In: *GeoDa Center for Geospatial Analysis and Computation, Arizona State University, Working Paper Series* 13 (2008), pp. 1–5. URL: *https://geodacenter.asu.edu/category/public/scalable-tile-m*.

[She19] Sheffield University. *Flexible Large Scale Agent Modelling Environment for the GPU*. *http://www.flamegpu.com*. 2019.

[She68] Donald Shepard. “A two-dimensional interpolation function for irregularly-spaced data”. In: *Proceedings of the 1968 23rd ACM National Conference*. ACM ’68. New York, NY, USA: ACM, 1968, pp. 517–524. DOI: *10.1145/800186.810616*. URL: *http://doi.acm.org/10.1145/800186.810616*.

[SK78] I. de Sola Pool and M. Kochen. “Contacts and influence”. In: *Social Networks* 1 (1978), pp. 5–51.

[Slo12] Mac Slocum, ed. *Big Data Now*. O’Reilly Media, 2012.

[SM08] A. Steed and R. Milton. “Using Tracked Mobile Sensors to Make Maps of Environmental Effects”. In: *Personal and Ubiquitous Computing* 12 (2008), pp. 331–342. DOI: *10.1007/s00779-006-0104-5*.

[SMC14] Christopher Smith-Clarke, Afra Mashhadi, and Licia Capra. “Poverty on the Cheap: Estimating Poverty Maps Using Aggregated Communication Networks”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 511–520. ISBN: 978-1-4503-2473-1. DOI: *10.1145/2556288.2557358*. URL: *http://doi.acm.org/10.1145/2556288.2557358*.

[Smi82] Alvy Ray Smith. “The Genesis Demo: Instant Evolution with Computer Graphics”. In: *American Cinematographer* 63.10 (Oct. 1982), pp. 1038–1050.

[Smi84] Alvy Ray Smith. “Plants, Fractals, and Formal Languages”. In: *SIGGRAPH Comput. Graph.* 18.3 (Jan. 1984), pp. 1–10. ISSN: 0097-8930. DOI: *10.1145/964965.808571*. URL: *http://doi.acm.org/10.1145/964965.808571*.

[SP98] K. M. Sivakholundu and N. Parabaharan. “A Program to Compute the Area of an Irregular Polygon on a Spheroidal Surface”. In: *Computers and Geosciences* 24.8 (1998), pp. 823–826.

[Ste+04] Anthony Steed et al. “Data Visualization within Urban Models”. In: *Theory and Practice of Computer Graphics, 2004*. Bournemouth: IEEE, June 2004, pp. 9–16. DOI: *10.1109/TPCG.2004.1314447*.

[Ste80] David Stevenson. “A Report on the Proposed IEEE Floating Point Standard (IEEE Task P754)”. In: *SIGARCH Comput. Archit. News* 8.15 (Aug. 1980), pp. 11–12. ISSN: 0163-5964. DOI: *10.1145/859510.859513*. URL: *http://doi.acm.org/10.1145/859510.859513*.

[SV89] John P. Snyder and M. Voxland Philip. “An Album of Map Projections”. In: *U.S. Geological Survey Professional Paper 1453* (1989).

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *CoRR* abs/1409.3215 (2014). arXiv: *1409.3215*. URL: *http://arxiv.org/abs/1409.3215*.

[SWK03] Kevin Sahr, Denis White, and A. Jon Kimerling. “Geodesic Discrete Global Grid Systems”. In: *Cartography and Geographic Information Science* 30.2 (2003), pp. 121–134.

[TLC02] F. Tecchia, C. Loscos, and Y. Chrysanthou. “Visualizing Crowds in Real-Time”. In: *Eurographics* 21.4 (2002), pp. 753–765.

[TM69] Jeffrey Travers and Stanley Milgram. “An Experimental Study of the Small World Problem”. In: *Sociometry* 32.4 (Dec. 1969), pp. 425–443. URL: *https://www.jstor.org/stable/2786545*.

[Tob04] W. Tobler. “Thirty Five Years of Computer Cartograms”. In: *Annals of the Association of American Geographers* 94 (2004), pp. 58–73. DOI: *10.1111/j.1467-8306.2004.09401004.x*.

[Tra14a] Transport for London. *CountDown API, TfL Developers Section*. 2014. URL: *http://www.tfl.gov.uk/businessandpartners/syndication/16493.aspx#21642*.

[Tra14b] Transport for London. *Transport for London Network Status & Train Prediction Services Development Beta SDK v0.2*. TfL (Transport for London). 2014. URL: *http://www.tfl.gov.uk/cdn/static/cms/documents/trackernet-data-services-guide-beta.pdf*.

[Tuf83] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[Ulr+18] Hannes Ulrich et al. “Using graph tools on metadata repositories”. In: Studies in Health Technology and Informatics 253 (2018), pp. 55–59. DOI: *10.3233/978-1-61499-896-9-55*.

[Ulr02] Thatcher Ulrich. “Rendering Massive Terrains using Chunked Level of Detail Control (DRAFT)”. (Published online). 2002. URL: *http://tulrich.com/geekstuff/sig-notes.pdf*.

[Uni14] Univeristy of Waikato. *MOA: Massive Online Analysis*. University of Waikato. 2014. URL: *http://moa.cms.waikato.ac.nz/*.

[Uno+09] Nobuhiro Uno et al. “Using Bus Probe Data for Analysis of Travel Time Variability”. In: *Journal of Intelligent Transportation Systems* 13.1 (2009), pp. 2–15. eprint: *https://doi.org/10.1080/15472450802644439*. URL: *https://doi.org/10.1080/15472450802644439*.

[Vie+07] F. B. Viegas et al. “Many Eyes: a Site for Visualization at Internet Scale”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (Nov. 2007), pp. 1121–1128. ISSN: 1077-2626. DOI: *10.1109/TVCG.2007.70577*.

[Viv13] Vivid Solutions. *Java Topology Suite (JTS)*. 2013. URL: *http://www.vividsolutions.com/jts/JTSHome.htm*.

[W3C05] W3C. *OWL Web Ontology Language*. 2005. URL: *http://www.w3.org/TR/2004/REC-owl-features-20040210/,2004*.

[Wac03] Hans Wackernagel. “Geostatistical models and kriging”. In: *IFAC Proceedings Volumes* 36.16 (2003), pp. 543–548. ISSN: 1474-6670. DOI: *https://doi.org/10.1016/S1474-6670(17)34818-18*. URL: *http://www.sciencedirect.com/science/article/pii/S1474667017348188*.

[Wan10] Shaowen Wang. “A CyberGIS Framework for the Synthesis of Cyberinfrastructure, GIS, and Spatial Analysis”. In: *Annals of the Association of American Geographers* 100:3 (2010), pp. 535–557. DOI: *10.1080/00045601003791234*.

[Wan15] Jiaqiu Wang. “Resilience of Self-Organised and Top-Down Planned Cities—A Case Study on London and Beijing Street Networks”. In: *PLOS ONE* 10.12 (Dec. 2015), pp. 1–20. DOI: *10.1371/journal.pone.0141736*. URL: *https://doi.org/10.1371/journal.pone.0141736*.

[War85] Daniel Wartenberg. “Multivariate Spatial Correlation: A Method for Exploratory Geographical Analysis”. In: *Geographical Analysis* 17.4 (Oct. 1985), pp. 263–283.

[WEH11] Ian H. Witten, Frank Eibe, and Mrak A. Hall. *Data Mining, Practical Machine Learning Tools and Techniques*. 3rd ed. Morgan Kaufman, Feb. 2011.

[Wei14] Eric W. Weisstein. *Oblate Spheroid*. MathWorld - A Wolfram Web Resource. 2014. URL: *http://mathworld.wolfram.com/OblateSpheroid.html*.

[Wel62] B. P. Welford. “Note on a Method for Calculating Corrected Sums of Squares and Products”. In: *Technometrics* 4.3 (Aug. 1962), pp. 419–420. URL: *http://www.jstor.org/stable/1266577*.

[Wen08] K. Wenbin Fang. *Parallel Data Mining on Graphics Processors*. GPU Miner, Google Code. Oct. 2008. URL: *http://gpuminer.googlecode.com/gpuminer.pdf*.

[Wes+14] Robert West et al. "Processing and Rendering Massive 3D Geospatial Environments using WebGL - The examples of OpenWebGlobe and SmartMobileMapping". In: 2014. URL: *https://pdfs.semanticscholar.org/662c/64531aab954c0232b94d96e5c51b31a6612f.pdf*.

[Wie11] Nancy Wiegrand. "INTEROP NETWORK TO SUPPORT GEOSPATIAL DATA SEMANTIC INTEROPERABILITY". In: Milwaukee, Wisconsin, May 2011. URL: *%5Curl%7Bhttps://researchgate.net/publication/267247078_INTEROP_NETWORK_TO_SUPPORT_DATA_SEMANTIC_INTEROPERABILITY%7D*.

[Wil99] Uri Wilensky. *NetLogo*. Center for Connected Learning and Computer-Based Modelling. 1999. URL: *http://ccl.northwestern.edu/netlogo/docs*.

[Wor15] World Meteorological Organisation. *WMO Publication 306 - Manual on Codes, Volume I*. World Meteorological Organisation. 2015. URL: *http://www.wmo.int/pages/prog/www/WMOCodes/WMO306_vl1/VolumeI.1.html*.

[WPH04] Yalin Wang, Ishin T. Phillips, and Robert M. Haralick. "Table structure understanding and its performance evaluation". In: *Pattern Recognition* 37.7 (2004), pp. 1479–1497. ISSN: 0031-3203. DOI: *https://doi.org/10.1016/j.patcog.2004.01.012*. URL: *http://www.sciencedirect.com/science/article/pii/S0031320304000494*.

[WR07] Uri Wilensky and William Rand. "Making Models Match: Replicating an Agent-Based Model". In: *Journal of Artificial Societies and Social Simulation* 10.4 (2007), p. 2. ISSN: 1460-7425. URL: *http://jasss.soc.surrey.ac.uk/10/4/2.html*.

[Wri38] John Kirtland Wright. "Problems in Population Mapping". In: *Notes on Statistical Mapping, with Reference to the Mapping of Population Phenomena.* Mimeographed Publication no. 1 (1938), pp. 1–18.

[WSB03] M. Walter, L. Stuart, and R. Borisyuk. "A Compact Visualisation for Neurophysiological Data". In: *Seventh International Conference on Information Visualization (IV03)* (2003). URL: *http://www.tech.plymouth.ac.uk/infovis/papers/walter_stuart_borisyuk_iv03_paper.pdf*.

[WWT99] David H. Wolpert, Kevin R. Wheeler, and Kagan Tumer. "General Principles of Learning-Based Multi-Agent Systems". In: *CoRR* cs.MA/9905005 (1999). URL: *http://arxiv.org/abs/cs.MA/9905005*.

[Yau11] Nathan Yau. *Visualize This*. Wiley Publishing Inc., 2011. URL: *flowingdata.com*.

[Yau13] Nathan Yau. *Data Points*. John Wiley & Sons, 2013.

[Zah+11] Matei Zaharia et al. "The datacenter needs an operating system". In: *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing*. USENIX Association. 2011, pp. 17–17.

[Zip49] G. K. Zipf. *Human Behaviour and the Principle of Least Effort*. Cambridge 42, Massachusetts: Addison-Wesley Press Inc., 1949.

Glossary

A

ABM	Agent Based Modelling.
AgentScript	Agent Based Modelling system written in Javascript and based loosely on the NetLogo implementation.
ArcGIS	GIS software created by ESRI.
ASXX	Met Office surface pressure actuals chart containing isobars and frontal positions.
AURN	Automatic Urban and Rural Network. Air quality monitoring sites in the UK.

B

BADC	British Atmospheric Data Centre. Archived weather and climate observations from the UK Meteorological Office available to researchers.
BOW	Bag of Words. Representation of a document based on the individual words that it contains without retaining any of the ordering. Used primarily in computer analysis of natural language.
Big O notation, $\mathcal{O}(N^2)$	Defines the order of an algorithm. For example $\mathcal{O}(N^2)$ means that the computation time until the algorithm stops scales in relation to N^2 where N counts the number of computations. N might be the number of areas in a map, or the number of integers input to a sorting algorithm.

C

C#	C Sharp, .net programming language created by Microsoft.
C++	Programming language created by Bjarne Stroustrup in 1979.
CASA	Centre for Advanced Spatial Analysis at University College London.
Choropleth	Map where areas are coloured according to a data value.
CLR	Common Language Runtime. This is the common format used to store Microsoft's compiled .net code before the generation of the executable code.
COLLADA	Format for storing 3D assets in the computer graphics industry. Largely superseded by FBX.
Countdown	System used by TfL for tracking buses in London.

CUDA	NVIDIA's technology for running general purpose code on their GPU architectures. OpenCL is the open alternative that also works on Radeon hardware.
CyberGIS	Cyberinfrastructure is a term for infrastructure that is spread across the Internet, so CyberGIS is GIS specifically focussed on distributed GIS infrastructure.

D

D3	Javascript library for web-based visualisations of data. D3 stands for "Data Driven Documents".
DirectX	Graphics library created by Microsoft for high performance 3D graphics on their operating systems. This includes commercial game development on the XBox platform.

E

EA	Environment Agency. Public body in the UK responsible for hydrology.
Edge Computing	Architecture where a request is satisfied at the edge of the network, preventing a reqeust for processing to propagate through to the application servers at the core. For example, in tiled maps, an edge cache containing previously rendered map tiles reduces the workload on the applications servers, which are only used to render tiles when there is a cache miss.
EQUATOR Project	Multi-disciplinary eScience project which collected carbon monoxide data in the field using GPS tracked sensors and visualised it on a 3D view of the city.
ESRI	Environmental Systems Research Institute, the organisation that created the ArcGIS platform.

F

FBX	Graphics interchange format pioneered by Sony for the import and export of assets in the computer games industry. The format is an XML markup.
FPU	Floating point unit. The hardware inside a computer that performs mathematical functions on floating point numbers, for example, addition, subtraction, multiplication, division, square root and trigonometric functions.
FSF	Free Software Foundation. Organisation responsible for the open source GNU software tools. Founded by Richard Stallman.
FSXX	Met Office forecast surface pressure chart containing predicted isobars and frontal positions.
Fusion Tables	Cloud based system run by Google which allows users to make maps from tables of geographically referenced data.

G

GenESiS Project	Generative eSocial Science project which combined modelling with web-based map visualisation.
GeoAPI	Set of interfaces published as an open standard to allow developers of GIS libraries to implement functions in a standardised way, offering interoperability between libraries.

GeoJSON	JSON format for storing geographic data in the form of points, lines and polygons with attributes.
GeoVUE Project	Geographic Virtual Urban Environments project which built tools to put data onto the newly released Google Maps.
Geotools	Open source Java library for GIS.
GEOS	C++ library for geometric manipulation. Forms the basis of numerous GIS projects developed in C++.
Gephi	Open source graph visualisation library written in Java.
GIS	Geographic Information System. Software for performing spatial manipulations on data.
GitHub	Website for storing code under version and managing projects with multiple developers with parallel working.
GMapCreator	Software created by CASA in 2006 for taking shapefile data and putting it onto a Google Map.
GML	Geography Markup Language. OGC standard for describing geographic features using an XML grammar.
GPGPU	General purpose programming with Graphics Processing Units.
GPU	Graphics Processing Unit, specialised microprocessor designed for rendering complex 3D graphics, but the high number of parallel cores can also be used for general purpose processing.
GTFS	General Transit File Specification. Created by Google for sharing public transport timetables. The format is text based, using CSV files for stop point locations and routes.

H

HadCRUT4	Global temperature data set containing global gridded temperatures from 1850 to 2018. Produced by the Uk Met Office's Hadley Centre in conjunction with Climatic Research Unit of the University of East Anglia.
Hadoop	Apache project and open source implementation of MapReduce.
HDFS	Hadoop distributed file system. The distributed file system underlying Hadoop which balances file storage across multiple servers.
Hyper-V	Microsoft virtualisation architecture.

I

I, Moran's I	Moran's I is a statistical measure of spatial autocorrelation.
Infinite Provisioning	In cloud computing, infinite provisioning is the phrase used to describe the fact that computing resource is rented out from the spare pool owned and run by the big Internet companies. While not actually infinite, smaller companies can buy as much compute resource as they need and can afford.
ISA	Instruction Set Architecture. An abstract representation of a computer architecture defining everything that a machine code programmer needs to write programs. This would normally include definitions of instructions, registers, memory and any other hardware or operations peculiar to the architecture.

J

Java	Programming language which is interpreted, or just in time complied, making it portable across operating systems and computing architectures.
Javascript	Programming language supported by web browsers.
JDBC	Java Database Connection. Protocol used by java to connect to databases using drivers written each database engine.
JIT	Just in time. Often used in conjunction with Java Virtual Machines, Just In Time compilation is a technique for speeding up interpreted code.
JSON	Javascript Object Notation. Format for storing data in Javascript readable files.
JTS	Java Topology Suite. Library for manipulating geometry, forming the base of many GIS applications. The .net version is NTS.
JVM	Java Virtual Machine. Program used to execute the bytecode produced by Java compilers. The ability to run the same bytecode on a different implementation of a JVM on another architecture is what gives Java its platform independence.

K

K-Nearest Neighbours	Spatial analysis technique based for classifying based on the 'k' closest areas, where k is an integer e.g. 3 closest, 4 closest etc.
KML	Keyhole Markup Language. XML markup for geographic data created by Google and based on the OGC's GML. Originally used to add vector data as overlays on Google Earth, it was given to the OGC and has since become one of their standards.

L

Leaflet	Lightweight Javascript maps library.
LRU	Least recently used. Scheme for clearing old data out of caches based on deleting things which have not been used for longest first.
LSOA	Lower layer Super Output Area, UK geography comprising 34,753 areas for England and Wales.

M

MapBoxGL	Open source library for tiled maps using WebGL to render vector tiles on the client.
MapTube	"A place to put maps". Website created in 2008 to make it easier for the general public to create maps and share them online.
MapTubeD	Tile renderer created as part of the MapTube family of projects.
MapReduce	Fundamental to cloud computing. The ability to map Big Data across thousands of servers and then reduce the aggregate results back to a single point.
Matrox G200	Reference 2D graphics card used in numerous server architectures.

Metar	Meteorological Aerodrome Report. Aviation synoptic observation for an airfield.
MIMD	Multiple Instruction Multiple Data stream. One of Flynn's four computer architectures [Fly66]. Multiple instruction pipelines act in parallel on multiple streams of data.
MISD	Multiple Instruction Single Data stream. One of Flynn's four computer architectures [Fly66].
MSIL	Microsoft Intermediate Language. This is the common format that all .net languages compile to before conversion into executable code. It can be likened to the output of a conventional compiler just before the final code generation stage.
MSOA	Middle layer Super Output Area, UK geography comprising 7,201 areas for England and Wales.

N

NCAR	National Center for Atmospheric Research in the United States.
NCeSS Project	National Centre for eSocial Science.
NCRM	National Centre for Research Methods.
NeISS Project	National e-infrastructure for Social Science. Project involving a group of universities which was funded by ESRC.
.Net Framework	Framework created by Microsoft where languages like C#, Visual Basic and Iron Python compile into an intermediate language (MSIL) which is then run using a common .net implementation. This makes it platform independent.
NetLogo	Implementation of an Agent Based Modelling framework in the Logo programming language. The environment has been developed in Java, making it portable.
NLP	Natural Language Processing. Term describing the analysis of human language using computer algorithms.
NOAA	National Oceanic and Atmospheric Administration in the United States.
NTS	Net Topology Suite. Port of the Java Topology Suite (JTS) to .net.

O

OA	Output Area, UK geography comprising 181,408 areas for England and Wales.
OGC	Open Geospatial Consortium.
ONS	Office for National Statistics, body in the UK that manages official Government statistics.
OpenCL	Open Compute Language. Allows general purpose code to be run on the GPU, independently of whether the GPU is an NVIDIA one, Radeon, or any other compliant hardware.
OpenGL	Open Graphics Language. Graphics framework that has been implemented on a wide variety of hardware over its lifetime, providing a degree of platform independence for developers writing 3D applications. Microsoft offer the DirectX library as a proprietary alternative.

OpenML	Open machine learning library.
OpenStreetMap	Open source geographic data covering the entire world.
Ordnance Survey	Body in the UK responsible for mapping.

P

Pattern (programming)	Software engineering methodology based on recognising recurring structures in software that are implemented as generalised patterns and re-used. For example, class 'Factories', as used in the Geotools library, are a common design pattern.
Protocol Buffers	Data format created by Google and used as the base format for MapBox vector tiles on their tiled maps.
Proxy	System whereby a request is sent to another machine first (the proxy), before being sent to its final destination. One use of a proxy server is to support a local cache.
PySAL	Python spatial analysis library.

Q

Quad tree segmentation	Algorithm for splitting a region into four recursively and so quadrupling the number of areas in each subsequent split. Used in web-based tiled maps where a single tile represents the whole world at the first zoom level. This then splits into four, then sixteen, then sixty four.
QUANT	Quantitative Analytics. Website for spatial interaction modelling, created by CASA. Uses MapBox vector tiles to achieve the complex GIS functions needed to display the outputs from the spatial interaction model.
QGIS (Quantum GIS)	Open source GIS software which is an alternative to the commercial ArcGIS package.

R

RAID	Redundant array of inexpensive disks. Hardware for managing an array of disks which appears as a single logical volume to the operating system. Depending on the RAID level, the array can be constructed to have redundancy in the case of single or double disk failures, or sacrifice fault tolerance for speed if 'disk striping' is used.
RDF	Resource Description Framework. W3C standard for storing linked semantic data as triples of subject, predicate, object.
REST	Representational state transfer. Method for requesting documents or processing using the URL to encode the request rather an a parameter string. The paradigm is based on the result of the operation being represented as an object or document on the Internet, rather than an operation.
Ruby	Programming language. Developed in the 1990s and made popular for web programming with the 'Ruby on Rails' open source web application framework.

S

Shapefile	Geographic file format made popular by the Environmental Systems Research Institute (ESRI).

SIMD	Single Instruction Multiple Data stream. One of Flynn's four computer architectures [Fly66]. One instruction pipeline acts in parallel on multiple streams of data. An example would be performing the same instruction on every element in a vector of data.
SISD	Single Instruction Single Data stream. One of Flynn's four computer architectures [Fly66]. One instruction pipeline acts on a single stream of data. This is the basic fundamental computing architecture with no parallelism.
SPARQL	SPARQL Protocol and RDF Query Language. Used for querying data from triple stores in semantically linked data.
Stemming Algorithm	In natural language processing, stemming reduces a word to its unique stem, so plural and possessives get reduced to a common root. For example, "computer", "computers" and "computer's" all get reduced to "comput". Porter's stemming algorithm contains the full dictionary [Por80].
Superscalar	Describes a microprocessor design where more than one instruction can be executed per clock cycle.
SurveyMapper	Website for crowd-sourcing geographic surveys: *https://www.surveymapper.com*.
SYMAP	GIS system developed by the Harvard Laboratory for Computer Graphics in 1965 and widely believed to be the first GIS.
Synop	Synoptic observation. Actual, observed, weather data for a specific location and time.

T

Talisman	The Talisman project was a node of the National Centre for Research Methods and concerned geographic data mining.
TOC	Train Operating Company. Acronym for a company running a train service in the UK.
TopoJSON	Similar to GeoJSON, but designed to be a more compressed and efficient format for storing geographic data in JSON format.
TfL	Transport for London. Body responsible for all transport systems running in London.
Trackernet	System used by TfL for tracking London Underground trains.

U

UML	Unified Modelling Language. Standard used in software engineering for the formal documentation of software.

V

VM, Virtual Machine	A virtual machine is an implementation of an operating system running as a program on a physical computer. The Hypervisor program, for example Microsoft Hyper-V, is the host program used to run multiple virtual machines on a single physical machine. The VM is termed the guest machine.

VLIW — Very Long Instruction Word. Computer architecture where a single machine instruction performs multiple functions. An example includes GPS instructions sets which control multiple parallel functional units simultaneously for efficiency.

W

WebGL — High performance 3D graphics library for web browsers.

WMO — World Meteorological Organisation. Founded in 1950 to promote the free exchange of meteorological information between participating countries.

WFS — Web Feature Service. OGC specification for exchanging information about geographic features over the web.

WMS — Web Map Service. OGC specification for making map requests over the web and having the server render the whole map as an image and return it as a PNG or JPEG file.

WMTS — Web Map Tile Service. OGC specification for making tiled map requests over the web. Used for tiled map services like Google Maps, MapBox, OpenLayers and Leaflet.

X

XHTML — Web page that conforms to a well-formed XML document.

XML — eXtensible Markup Language. Text representation for storing data using entity tags and attributes that can be defined by the user and published as a schema in an XSD document.

XSD — XML Schema Definition. Formal language specifying the syntax of an XML document.

XSLT — eXtensible Stylesheet Language. Transformational language for XML documents that can convert from one schema into another. For example, an XSLT document is used by MapTube to transform a list of map layers into the html page seen by the user.

Y

YARN — Yet another resource negotiator. One of Apache Hadoop's core components for managing resources and job scheduling on a Hadoop cluster.

Z

Zipf's Law — Distribution pattern in data where the frequency of an item is inversely proportional to its ranking by frequency.

Appendices

This appendix contains references to all the sources of data used in the preceding chapters.

A NOMIS Census Bulk Upload r2.2

The NOMIS bulk upload of data from the 2011 Census is available at the following address: *https://www.nomisweb.co.uk/census/2011/bulk/r2_2*.

The list of tables is downloadable from the following address: *http://www.maptube.org/PhDData/CH5-DynamicVisualisation/NOMIS2011CensusBulk.csv*.

The list of variable codes, along with their descriptions is downloadable from the following address: *http://www.maptube.org/PhDData/CH5-DynamicVisualisation/NOMIS2011Variables.txt*.

B TfL Underground Station Codes

Code	Lines	Lon	Lat	Name
ACT	DP	-0.2803	51.5027	Acton Town
ALD	MH	-0.0756	51.5143	Aldgate
ALE	DH	-0.0723	51.5152	Aldgate East
ALP	P	-0.2995	51.5407	Alperton
AME	M	-0.6075	51.6741	Amersham
ANG	N	-0.1058	51.5325	Angel
ARC	N	-0.1351	51.5654	Archway
AGR	P	-0.1335	51.6163	Arnos Grove
ARL	P	-0.1058	51.5585	Arsenal
BST	MBHJ	-0.1569	51.5231	Baker Street
BAL	N	-0.1532	51.4433	Balham
BNK	WNC	-0.0889	51.5133	Bank
BAR	MH	-0.0977	51.5201	Barbican
BAY	DH	-0.1880	51.5122	Bayswater
BKG	DH	0.0809	51.5395	Barking
BDE	C	0.0885	51.5858	Barkingside
BCT	DP	-0.2136	51.4902	Barons Court
BEC	D	0.1274	51.5403	Becontree
BPK	N	-0.1642	51.5504	Belsize Park
BER	J	-0.0637	51.4979	Bermondsey
BNG	C	-0.0554	51.5272	Bethnal Green
BLF	DH	-0.1036	51.5115	Blackfriars
BHR	V	-0.0410	51.5870	Blackhorse Road
BDS	CJ	-0.1493	51.5138	Bond Street
BOR	N	-0.0937	51.5012	Borough
BOS	P	-0.3248	51.4957	Boston Manor
BGR	P	-0.1242	51.6071	Bounds Green
BWR	DH	-0.0248	51.5268	Bow Road
BTX	N	-0.2135	51.5768	Brent Cross
BRX	V	-0.1146	51.4626	Brixton
BBB	DH	-0.0117	51.5248	Bromley-by-Bow
BHL	C	0.0467	51.6265	Buckhurst Hill

Code	Lines	Lon	Lat	Name
BUR	N	-0.2642	51.6027	Burnt Oak
CRD	P	-0.1183	51.5484	Caledonian Road
CTN	N	-0.1427	51.5394	Camden Town
CWR	J	-0.0497	51.4979	Canada Water
CWF	J	-0.0194	51.5036	Canary Wharf
CNT	J	0.0082	51.5138	Canning Town
CST	DH	-0.0907	51.5114	Cannon Street
CPK	J	-0.2947	51.6078	Canons Park
CLF	M	-0.5605	51.6680	Chalfont and Latimer
CHF	N	-0.1537	51.5441	Chalk Farm
CYL	C	-0.1117	51.5181	Chancery Lane
CHX	BN	-0.1248	51.5086	Charing Cross
CHG	C	0.0745	51.6179	Chigwell
CHM	M	-0.6113	51.7052	Chesham
CHP	D	-0.2677	51.4943	Chiswick Park
CWD	M	-0.5184	51.6542	Chorleywood
CPC	N	-0.1383	51.4617	Clapham Common
CPN	N	-0.1295	51.4648	Clapham North
CPS	N	-0.1480	51.4526	Clapham South
CFS	P	-0.1496	51.6517	Cockfosters
COL	N	-0.2501	51.5953	Colindale
CLW	N	-0.1777	51.4181	Colliers Wood
COV	P	-0.1242	51.5129	Covent Garden
CRX	M	-0.4417	51.6470	Croxley
DGE	D	0.1659	51.5441	Dagenham East
DGH	D	0.1477	51.5416	Dagenham Heathway
DEB	C	0.0838	51.6454	Debden
DHL	J	-0.2388	51.5519	Dollis Hill
EBY	DC	-0.3015	51.5149	Ealing Broadway
ECM	DP	-0.2883	51.5101	Ealing Common
ECT	DP	-0.1935	51.4918	Earl's Court
EAC	C	-0.2475	51.5166	East Acton

Code	Lines	Lon	Lat	Name
EFY	N	-0.1647	51.5873	East Finchley
EHM	DH	0.0515	51.5389	East Ham
EPY	D	-0.2110	51.4588	East Putney
ETE	MP	-0.3968	51.5765	Eastcote
EDG	N	-0.2750	51.6136	Edgware
ERB	B	-0.1701	51.5202	Edgware Road (Bakerloo)
ERD	DH	-0.1677	51.5199	Edgware Road (H and C)
ELE	NB	-0.1007	51.4958	Elephant and Castle
EPK	D	0.1992	51.5498	Elm Park
EMB	DBNH	-0.1224	51.5072	Embankment
EPP	C	0.1139	51.6936	Epping
EUS	NV	-0.1333	51.5286	Euston
ESQ	MH	-0.1358	51.5256	Euston Square
FLP	C	0.0909	51.5957	Fairlop
FAR	MH	-0.1051	51.5204	Farringdon
FYC	N	-0.1924	51.6010	Finchley Central
FRD	MJ	-0.1805	51.5471	Finchley Road
FPK	PV	-0.1065	51.5644	Finsbury Park
FBY	D	-0.1950	51.4805	Fulham Broadway
GHL	C	0.0661	51.5765	Gants Hill
GRD	DPH	-0.1830	51.4942	Gloucester Road
GGR	N	-0.1940	51.5722	Golders Green
GST	N	-0.1347	51.5204	Goodge Street
GRH	C	0.0921	51.6133	Grange Hill
GPS	MH	-0.1440	51.5237	Great Portland Street
GPK	PVJ	-0.1429	51.5069	Green Park
GFD	C	-0.3464	51.5423	Greenford
GUN	D	-0.2752	51.4918	Gunnersbury
HAI	C	0.0931	51.6037	Hainault
HMS	H	-0.2249	51.4935	Hammersmith

Code	Lines	Lon	Lat	Name
HMD	DP	-0.2224	51.4926	Hammersmith (District and Picc)
HMP	N	-0.1782	51.5567	Hampstead
HLN	C	-0.2930	51.5300	Hanger Lane
HSD	B	-0.2575	51.5362	Harlesden
HAW	B	-0.3352	51.5922	Harrow and Wealdstone
HOH	M	-0.3370	51.5793	Harrow on the Hill
HTX	P	-0.4234	51.4666	Hatton Cross
HRF	P	-0.4461	51.4586	Heathrow Terminal 4
HRV	P	-0.488	51.4723	Heathrow Terminal 5
HRC	P	-0.4524	51.4712	Heathrow Terminals 123
HND	N	-0.2265	51.5833	Hendon Central
HBT	N	-0.1948	51.6506	High Barnet
HST	DH	-0.1925	51.5007	High Street Kensington
HBY	V	-0.1040	51.5462	Highbury and Islington
HIG	N	-0.1466	51.5776	Highgate
HDN	MP	-0.4499	51.5537	Hillingdon
HOL	CP	-0.1200	51.5174	Holborn
HPK	C	-0.2057	51.5073	Holland Park
HRD	P	-0.1129	51.5528	Holloway Road
HCH	D	0.2190	51.5540	Hornchurch
HNC	P	-0.3669	51.4711	Hounslow Central
HNE	P	-0.3567	51.4732	Hounslow East
HNW	P	-0.3857	51.4730	Hounslow West
HPC	P	-0.1527	51.5028	Hyde Park Corner
ICK	MP	-0.4420	51.5620	Ickenham
KEN	N	-0.1055	51.4881	Kennington
KGN	B	-0.2247	51.5305	Kensal Green
OLY	D	-0.2104	51.4978	Kensington (Olympia)
KTN	N	-0.1405	51.5503	Kentish Town
KNT	B	-0.3172	51.5818	Kenton
KEW	D	-0.2853	51.4770	Kew Gardens
KIL	J	-0.2046	51.5469	Kilburn

Code	Lines	Lon	Lat	Name
KPK	B	-0.1940	51.5351	Kilburn Park
KXX	MNPHV	-0.1239	51.5304	Kings Cross St. Pancras
KBY	J	-0.2788	51.5848	Kingsbury
KNB	P	-0.1605	51.5016	Knightsbridge
LAM	B	-0.1118	51.4991	Lambeth North
LAN	C	-0.1754	51.5118	Lancaster Gate
LBG	H	-0.2109	51.5172	Ladbroke Grove
LSQ	PN	-0.1282	51.5112	Leicester Square
LEY	C	-0.0056	51.5564	Leyton
LYS	C	0.0082	51.5682	Leytonstone
LST	MCH	-0.0830	51.5173	Liverpool Street
LON	NJ	-0.0869	51.5055	London Bridge
LTN	C	0.0553	51.6415	Loughton
MDV	B	-0.1856	51.5298	Maida Vale
MNR	P	-0.0957	51.5708	Manor House
MAN	DH	-0.0942	51.5120	Mansion House
MAR	C	-0.1584	51.5135	Marble Arch
MYB	B	-0.1631	51.5222	Marylebone
MLE	DHC	-0.0334	51.5251	Mile End
MHE	N	-0.2099	51.6083	Mill Hill East
MON	DH	-0.0889	51.5133	Monument
MPK	M	-0.4327	51.6297	Moor Park
MGT	MNH	-0.0890	51.5184	Moorgate
MOR	N	-0.1948	51.4023	Morden
MCR	N	-0.1388	51.5342	Mornington Crescent
NEA	J	-0.2498	51.5540	Neasden
NEP	C	0.0903	51.5756	Newbury Park
NAC	C	-0.2597	51.5234	North Acton Junction
NEL	P	-0.2890	51.5176	North Ealing
NGW	J	0.0036	51.5002	North Greenwich
NHR	M	-0.3622	51.5848	North Harrow
NWM	B	-0.3042	51.5625	North Wembley

Code	Lines	Lon	Lat	Name
NFD	P	-0.3142	51.4993	Northfields
NHT	C	-0.3685	51.5482	Northolt
NWP	M	-0.3182	51.5786	Northwick Park
NWD	M	-0.4239	51.6112	Northwood
NWH	M	-0.4093	51.6005	Northwood Hills
NHG	DCH	-0.1965	51.5091	Notting Hill Gate
NPDEPOT	V	-0.0538	51.5999	Northumberland Park Depot
OAK	P	-0.1318	51.6476	Oakwood
OLD	N	-0.0875	51.5256	Old Street
OST	P	-0.3520	51.4809	Osterley
OVL	N	-0.1129	51.4821	Oval
OXC	CBV	-0.1418	51.5151	Oxford Circus
PAD	HDB	-0.1754	51.5153	Paddington
PRY	P	-0.2842	51.5269	Park Royal
PGR	D	-0.2012	51.4751	Parsons Green
PER	C	-0.3239	51.5366	Perivale
PIC	BP	-0.1340	51.5100	Piccadilly Circus
PIM	V	-0.1337	51.4892	Pimlico
PIN	M	-0.3809	51.5929	Pinner
PLW	DH	0.0178	51.5312	Plaistow
PUT	D	-0.2090	51.4679	Putney Bridge
QPK	B	-0.2047	51.5341	Queen’s Park
QBY	J	-0.2858	51.5943	Queensbury
QWY	C	-0.1874	51.5104	Queensway
RCP	D	-0.2363	51.4941	Ravenscourt Park
RLN	MP	-0.3710	51.5750	Rayners Lane
RED	C	0.0454	51.5763	Redbridge
RPK	B	-0.1469	51.5235	Regents Park
RMD	D	-0.3018	51.4632	Richmond
RKY	M	-0.4737	51.6403	Rickmansworth
ROA	H	-0.1882	51.5190	Royal Oak
ROD	C	0.0439	51.6171	Roding Valley

Code	Lines	Lon	Lat	Name
RUI	MP	-0.4215	51.5714	Ruislip
RUG	C	-0.4110	51.5606	Ruislip Gardens
RUM	MP	-0.4123	51.5732	Ruislip Manor
RSQ	P	-0.1243	51.5229	Russell Square
SVS	V	-0.0725	51.5834	Seven Sisters
SBC	C	-0.2263	51.5056	Shepherd's Bush
XSBM	H	-0.2264	51.5058	Shepherd's Bush Market
SSQ	DH	-0.1565	51.4923	Sloane Square
SNB	C	0.0215	51.5808	Snaresbrook
SEL	P	-0.3070	51.5014	South Ealing
SHR	P	-0.3522	51.5647	South Harrow
SKN	DPH	-0.1739	51.4940	South Kensington
SKT	B	-0.3086	51.5702	South Kenton
SRP	C	-0.3987	51.5565	South Ruislip
SWM	N	-0.1920	51.4153	South Wimbledon
SWF	C	0.0273	51.5917	South Woodford
SFS	D	-0.2065	51.4449	Southfields
SGT	P	-0.1278	51.6323	Southgate
SWK	J	-0.1051	51.5038	Southwark
SJW	J	-0.1742	51.5346	St. John's Wood
STP	C	-0.0976	51.5148	St. Paul's
SJP	DH	-0.1342	51.4993	St. James's Park
STB	D	-0.2455	51.4948	Stamford Brook
STA	J	-0.3031	51.6196	Stanmore
STG	DH	-0.0465	51.5219	Stepney Green
STK	NV	-0.1229	51.4722	Stockwell
SPK	B	-0.2754	51.5439	Stonebridge Park
SFD	CJ	-0.0032	51.5413	Stratford
SHL	P	-0.3362	51.5570	Sudbury Hill
STN	P	-0.3155	51.5508	Sudbury Town
SWC	J	-0.1748	51.5433	Swiss Cottage

Code	Lines	Lon	Lat	Name
TEM	DH	-0.1137	51.5110	Temple
THB	C	0.1031	51.6717	Theydon Bois
TBE	N	-0.1597	51.4358	Tooting Bec
TBY	N	-0.1680	51.4274	Tooting Broadway
TCR	CN	-0.1309	51.5162	Tottenham Court Road
TTH	V	-0.0603	51.5880	Tottenham Hale
TOT	N	-0.1793	51.6302	Totteridge and Whetstone
THL	DH	-0.0764	51.5101	Tower Hill
TPK	N	-0.1379	51.5567	Tufnell Park
TGR	DP	-0.2545	51.4951	Turnham Green
TPL	P	-0.1028	51.5903	Turnpike Lane
UPM	D	0.2511	51.5589	Upminster
UPB	D	0.2358	51.5588	Upminster Bridge
UPY	D	0.1016	51.5383	Upney
UPK	DH	0.0353	51.5352	Upton Park
UXB	MP	-0.4781	51.5465	Uxbridge
VUX	V	-0.1237	51.4857	Vauxhall
VIC	DHV	-0.1438	51.4963	Victoria
WAL	V	-0.0199	51.5830	Walthamstow Central
WAN	C	0.0287	51.5755	Wanstead
WST	VN	-0.1383	51.5245	Warren Street
WAR	B	-0.1837	51.5233	Warwick Avenue
WLO	WBNJ	-0.1141	51.5035	Waterloo
WAT	M	-0.4173	51.6574	Watford
WEM	B	-0.2969	51.5523	Wembley Central
WPK	MJ	-0.2793	51.5633	Wembley Park
WAC	C	-0.2810	51.5179	West Acton
WBP	H	-0.2009	51.5209	Westbourne Park
WBT	D	-0.1955	51.4873	West Brompton
WFY	N	-0.1885	51.6095	West Finchley
WHM	DHJ	0.0050	51.5282	West Ham
WHD	J	-0.1907	51.5467	West Hampstead

Code	Lines	Lon	Lat	Name
WHR	J	-0.3529	51.5798	West Harrow
WKN	D	-0.2065	51.4905	West Kensington
WRP	C	-0.4379	51.5695	West Ruislip
WMS	DHJ	-0.1248	51.5011	Westminster
WCT	C	-0.2243	51.5120	White City
WCTSIDE	C	-0.2243	51.5120	White City Sidings
WCL	DH	-0.0600	51.5195	Whitechapel
WLG	J	-0.2224	51.5493	Willesden Green
WJN	B	-0.2443	51.5322	Willesden Junction
WDN	D	-0.2064	51.4214	Wimbledon
WMP	D	-0.1996	51.4345	Wimbledon Park
WGN	P	-0.1096	51.5975	Wood Green
WGNSIDE	P	-0.1096	51.5975	Wood Green Sidings
WFD	C	0.0340	51.6070	Woodford Junction
WFDSIDE	C	0.0340	51.6070	Woodford Sidings
WSP	N	-0.1854	51.6178	Woodside Park
XWLN	H	0.2242	51.5098	Wood Lane
XPRD	M	-0.2953	51.5720	Preston Road
XLRD	H	-0.2178	51.5135	Latimer Road
XGRD	H	-0.2267	51.5020	Goldhawk Road
PADc	H	-0.1774	51.5173	Paddington Circle
PADs	H	-0.1774	51.5173	Paddington H and C
XXX5	B	-0.2078	51.5336	North Sidings between Queen's park
XXX9	B	-0.2078	51.5336	Queen's Park North Sidings
XXX6	B	-0.1047	51.4983	London Road Depot
XXX8	V	-0.0199	51.5830	Walthamstow Sidings
XKRI	B	-0.2208	51.5342	Kensal Rise

C TfL Underground Destination Codes

The following file contains all the unique destination codes in use by TfL on the London Underground during January 2014. The data was produced by using a data minning algorithm on TfL's live tube API service: *http://www.maptube.org/PhDData/CH6-RealTimeMappingAndAgentSimulations/AllJan2014_UniqueDestCodes.csv*

A list of all destination codes processed for 2014 is available at the following link: *http://www.maptube.org/PhDData/CH6-RealTimeMappingAndAgentSimulations/trackernet_destinationcode_usage_jan2014_linecodes.csv*. This contains a list of every destination code seen on the public real-time API stream, along with the plain text for the destination, a count of the number of times it occurred and a line code where this can be uniquely identified. Where there is conflicting data, the line code is listed as '?' and the code is unidentified.

D AURN Air Quality Sensor Network

The following is a list of all the AURN air quality sensor sites in the UK.

latitude	longitude	site id	uka id	site name
57.1573	-2.0942	ABD	UKA00399	Aberdeen
57.1445	-2.1064	ABD7	UKA00513	Aberdeen Union Street Roadside
54.3537	-6.6545	ARM6	UKA00541	Armagh Roadside
52.5038	-3.0341	AH	UKA00137	Aston Hill
55.7921	-3.2429	ACTH	UKA00451	Auchencorth Moss
54.8615	-6.2508	BALM	UKA00503	Ballymena Ballykeel
53.5629	-1.5104	BAR3	UKA00353	Barnsley Gawber
51.0747	-4.0419	BPLE	UKA00574	Barnstaple A39
51.3911	-2.3541	BATH	UKA00306	Bath Roadside
54.5996	-5.9288	BEL2	UKA00212	Belfast Centre
54.5725	-5.9749	BEL1	UKA00594	Belfast Stockman's Lane
54.6053	-1.2750	BIL	UKA00153	Billingham
52.4371	-1.8299	AGRN	UKA00559	Birmingham Acocks Green
52.5117	-1.8305	BIR1	UKA00479	Birmingham Tyburn
52.5121	-1.8308	BIRT	UKA00543	Birmingham Tyburn Roadside
53.7477	-2.4527	BLAR	UKA00590	Blackburn Accrington Road
53.8048	-3.0071	BLC2	UKA00488	Blackpool Marton
52.9302	-0.8147	BOT	UKA00055	Bottesford
50.7395	-1.8267	BORN	UKA00429	Bournemouth
53.7712	-1.7597	BDMA	UKA00611	Bradford Mayo Avenue
50.8408	-0.1475	BRT3	UKA00483	Brighton Preston Park
51.4628	-2.5844	BRS8	UKA00494	Bristol St Paul's
53.5590	-2.2937	BURW	UKA00598	Bury Whitefield Roadside
55.8622	-3.2057	BUSH	UKA00128	Bush Estate
52.2023	0.1244	CAM	UKA00396	Cambridge Roadside
51.5442	-0.1752	CA1	UKA00259	Camden Kerbside
52.6747	-2.0308	CANR	UKA00597	Cannock Watling Street
51.2739	1.0980	CANT	UKA00424	Canterbury

latitude	longitude	site id	uka id	site name
51.4817	-3.1762	CARD	UKA00217	Cardiff Centre
54.8948	-2.9453	CARL	UKA00526	Carlisle Roadside
51.0562	-2.6834	MACK	UKA00537	Charlton Mackrell
51.3742	0.5479	CHAT	UKA00553	Chatham Roadside
51.6380	-2.6787	CHP	UKA00515	Chepstow A48
53.2441	-1.4549	CHLG	UKA00604	Chesterfield Loundsley Green
53.2317	-1.4569	CHS7	UKA00529	Chesterfield Roadside
52.4115	-1.5602	COAL	UKA00592	Coventry Allesley
51.6538	-3.0069	CWMB	UKA00436	Cwmbran
55.0012	-7.3291	DERY	UKA00343	Derry
53.5188	-1.1380	DCST	UKA00612	Doncaster A630 Cleveland Street
55.9431	-4.5597	DUMB	UKA00555	Dumbarton Roadside
55.0700	-3.6142	DUMF	UKA00428	Dumfries
51.5189	-0.2656	EA8	UKA00591	Ealing Horn Lane
50.8057	0.2716	EB	UKA00546	Eastbourne
55.9455	-3.1821	ED3	UKA00454	Edinburgh St Leonards
55.3153	-3.2061	ESK	UKA00130	Eskdalemuir
50.7250	-3.5324	EX	UKA00263	Exeter Roadside
56.8226	-5.1011	FW	UKA00495	Fort William
55.8720	-4.2709	GGWR	UKA00593	Glasgow Great Western Road
55.8609	-4.2382	GHSR	UKA00602	Glasgow High Street
55.8591	-4.2588	GLA4	UKA00336	Glasgow Kerbside
55.8657	-4.2436	GLKP	UKA00576	Glasgow Townhead
53.4600	-2.4720	GLAZ	UKA00170	Glazebury
56.0103	-3.7043	GRAN	UKA00420	Grangemouth
56.0131	-3.7108	GRA2	UKA00544	Grangemouth Moray
54.6842	-2.4507	GDF	UKA00134	Great Dun Fell
51.6805	-3.1335	CAE6	UKA00596	Hafod-yr-ynys Roadside
51.5993	-0.0682	HG1	UKA00260	Haringey Roadside
51.5710	-1.3252	HAR	UKA00047	Harwell
54.3349	-0.8085	HM	UKA00169	High Muffles
50.7922	-3.1967	HONI	UKA00566	Honiton

latitude	longitude	site id	uka id	site name
51.1658	-0.1677	HORE	UKA00511	Horley
53.7487	-0.3412	HUL2	UKA00450	Hull Freetown
53.7589	-0.3057	HULR	UKA00600	Hull Holderness Road
57.4813	-4.2414	INV2	UKA00434	Inverness
53.4033	-1.7520	LB	UKA00171	Ladybower
52.2888	-1.5331	LEAM	UKA00265	Leamington Spa
52.2948	-1.5429	LEAR	UKA00564	Leamington Spa Rugby Road
53.8037	-1.5464	LEED	UKA00222	Leeds Centre
53.8199	-1.5763	LED6	UKA00527	Leeds Headingley Kerbside
52.6386	-1.1242	LEIR	UKA00609	Leicester A594 Roadside
52.6198	-1.1273	LECU	UKA00573	Leicester University
52.2217	-2.7366	LEOM	UKA00489	Leominster
60.1392	-1.1853	LERW	UKA00486	Lerwick
53.2213	-0.5341	LIN3	UKA00561	Lincoln Canwick Rd.
53.4469	-2.9625	LV6	UKA00517	Liverpool Queen's Drive Roadside
53.3463	-2.8443	LVP	UKA00247	Liverpool Speke
51.4660	0.1848	BEX	UKA00238	London Bexley
51.5222	-0.1258	CLL2	UKA00211	London Bloomsbury
51.4525	0.0707	LON6	UKA00230	London Eltham
51.5841	-0.1252	HG4	UKA00568	London Haringey Priory Park South
51.4887	-0.4416	HRL	UKA00472	London Harlington
51.6173	-0.2987	HR3	UKA00540	London Harrow Stanmore
51.4963	-0.4608	HIL	UKA00266	London Hillingdon
51.5225	-0.1546	MY1	UKA00315	London Marylebone Road
51.5210	-0.2134	KC1	UKA00253	London N. Kensington
51.4209	-0.3396	TED	UKA00267	London Teddington
51.4252	-0.3456	TED2	UKA00572	London Teddington Bushy Park
51.4946	-0.1319	HORS	UKA00435	London Westminster
54.4395	-7.9003	LN	UKA00166	Lough Navar

latitude	longitude	site id	uka id	site name
50.7937	0.1812	LH	UKA00152	Lullington Heath
51.8922	-0.4621	LUTR	UKA00605	Luton A505 Roadside
53.3264	-9.9039	MH	UKA00167	Mace Head
53.4815	-2.2378	MAN3	UKA00248	Manchester Piccadilly
53.3690	-2.2432	MAN4	UKA00313	Manchester South
52.5544	-0.7722	MKTH	UKA00463	Market Harborough
54.5692	-1.2208	MID	UKA00220	Middlesbrough
51.7817	-4.6914	PEMB	UKA00323	Narberth
54.9782	-1.6105	NEWC	UKA00213	Newcastle Centre
54.9864	-1.5953	NCA3	UKA00528	Newcastle Cradlewell Roadside
51.6012	-2.9772	NPT3	UKA00380	Newport
52.2718	-0.8798	NTN3	UKA00567	Northampton Kingsthorpe
52.6141	1.3019	NO12	UKA00549	Norwich Lakenfields
52.9547	-1.1464	NOTT	UKA00274	Nottingham Centre
52.5024	-2.0034	BOLD	UKA00595	Oldbury Birmingham Road
51.7517	-1.2574	OX	UKA00258	Oxford Centre Roadside
51.7448	-1.2602	OX8	UKA00518	Oxford St Ebbes
55.6574	-3.1965	PEEB	UKA00551	Peebles
50.3716	-4.1423	PLYM	UKA00360	Plymouth Centre
51.5839	-3.7708	PT4	UKA00501	Port Talbot Margam
50.8288	-1.0685	PMTH	UKA00421	Portsmouth
53.7655	-2.6803	PRES	UKA00408	Preston
51.4530	-0.9440	REA1	UKA00462	Reading New Town
51.4561	0.6348	ROCH	UKA00251	Rochester Stoke
53.4848	-2.3341	ECCL	UKA00339	Salford Eccles
50.4114	-4.2276	SASH	UKA00569	Saltash Callington Road
52.1324	-0.3003	SDY	UKA00533	Sandy Roadside
53.5863	-0.6368	SCN2	UKA00381	Scunthorpe Town
53.5792	-2.0937	CW	UKA00579	Shaw Crompton Way
53.3786	-1.4780	SHDG	UKA00575	Sheffield Devonshire Green
53.4105	-1.3961	SHE	UKA00181	Sheffield Tinsley

latitude	longitude	site id	uka id	site name
52.2944	1.4634	SIB	UKA00012	Sibton
50.9081	-1.3957	SOUT	UKA00235	Southampton Centre
51.5442	0.6784	SEND	UKA00409	Southend-on-Sea
51.4804	-0.0595	SK5	UKA00558	Southwark A2 Old Kent Road
51.7779	1.0490	OSY	UKA00445	St Osyth
51.5181	0.4395	HOPE	UKA00525	Stanford-le-Hope Roadside
54.5658	-1.3159	SOTR	UKA00599	Stockton-on-Tees A1305 Roadside
54.5166	-1.3585	EAGL	UKA00535	Stockton-on-Tees Eaglescliffe
52.9804	-2.1118	STKR	UKA00610	Stoke-on-Trent A50 Roadside
53.0282	-2.1751	STOK	UKA00337	Stoke-on-Trent Centre
50.9169	-0.4495	STOR	UKA00548	Storrington Roadside
57.7344	-4.7765	SV	UKA00162	Strathvaich
54.8836	-1.4068	SUN2	UKA00484	Sunderland Silksworth
54.9183	-1.4083	SUNR	UKA00601	Sunderland Wessington Way
51.6326	-3.9473	SWA1	UKA00497	Swansea Roadside
51.4770	0.3179	THUR	UKA00272	Thurrock
51.5225	-0.0421	TH2	UKA00257	Tower Hamlets Roadside
52.6056	-2.0305	WAL4	UKA00565	Walsall Woodlands
53.3892	-2.6153	WAR	UKA00538	Warrington
52.9504	1.1220	WEYB	UKA00433	Weybourne
52.2985	0.2909	WFEN	UKA00362	Wicken Fen
53.3653	-2.7316	WSMR	UKA00603	Widnes Milton Road
53.5491	-2.6381	WIG5	UKA00482	Wigan Centre
53.3728	-3.0227	TRAN	UKA00406	Wirral Tranmere
53.0422	-3.0027	WREX	UKA00440	Wrexham
50.5976	-3.7165	YW	UKA00168	Yarner Wood
53.9675	-1.0865	YK10	UKA00523	York Bootham
53.9518	-1.0758	YK11	UKA00524	York Fishergate
53.2305	-1.4336	CHS6	UKA00530	Chesterfield
53.7155	-2.4838	BLCB	UKA00545	Blackburn Darwen Roadside
52.3943	-1.5196	COV3	UKA00427	Coventry Memorial Park

E Meteorological Office Datapoint UK Observation Stations

The following is a list of UK Meteorological Office observing stations reported on the 'Datapoint' API.

Name	WMO Number	Latitude	Longitude
BALTASOUND	3002	60.7490	-0.8539
LERWICK (S. SCREEN)	3005	60.1389	-1.1829
FAIR ISLE	3008	59.5270	-1.6280
KIRKWALL	3017	58.9539	-2.9000
SOUTH UIST RANGE	3023	57.3580	-7.3969
STORNOWAY	3026	58.2140	-6.3249
LOCH GLASCARNOCH SAWS	3031	57.7249	-4.8959
AULTBEA	3034	57.8590	-5.6360
SKYE/LUSA (SAMOS)	3037	57.2569	-5.8090
BEALACH NA BA	3039	57.4174	-5.6890
ALTNAHARRA SAWS	3044	58.2879	-4.4419
TULLOCH BRIDGE	3047	56.8670	-4.7080
TAIN RANGE	3062	57.8190	-3.9660
AVIEMORE	3063	57.2060	-3.8269
CAIRN GORM SUMMIT	3065	57.1160	-3.6419
KINLOSS	3066	57.6493	-3.5606
LOSSIEMOUTH	3068	57.7120	-3.3220
CAIRNWELL	3072	56.8790	-3.4200
WICK AIRPORT	3075	58.4539	-3.0889
ABOYNE	3080	57.0769	-2.8359
INVERBERVIE	3088	56.8520	-2.2639
ABERDEEN DYCE	3091	57.2060	-2.2019
TIREE	3100	56.4970	-6.8870
ISLAY/PORT ELLEN	3105	55.6809	-6.2560
MACHRIHANISH	3111	55.4410	-5.6989
WEST FREUGH (ESAWS)	3132	54.8590	-4.9359
GLASGOW/BISHOPTON	3134	55.9070	-4.5329
PRESTWICK RNAS	3136	55.5149	-4.5850

Name	WMO Number	Latitude	Longitude
STRATHALLAN	3144	56.3260	-3.7290
GLEN OGLE	3148	56.4230	-4.3200
DUNDRENNAN	3153	54.8030	-4.0079
DRUMALBIN	3155	55.6269	-3.7349
CHARTERHALL	3158	55.7089	-2.3829
ESKDALEMUIR	3162	55.3110	-3.2060
EDINBURGH/GOGARBANK	3166	55.9280	-3.3429
LEUCHARS	3171	56.3769	-2.8619
RONALDSWAY	3204	54.0848	-4.6321
ST. BEES HEAD	3210	54.5180	-3.6150
KESWICK	3212	54.6139	-3.1570
WALNEY ISLAND	3214	54.125	-3.2569
CARLISLE	3220	54.9329	-2.9630
SPADEADAM	3224	55.0499	-2.5529
SHAP	3225	54.5009	-2.6840
WARCOP	3226	54.5719	-2.4130
GREAT DUN FELL 2	3227	54.6839	-2.4500
REDESDALE CAMP (SAMOS)	3230	55.2849	-2.2790
ALBEMARLE	3238	55.0200	-1.8799
BOULMER	3240	55.4210	-1.6000
LEEMING	3257	54.2960	-1.5299
DISHFORTH AIRFIELD	3261	54.1339	-1.4140
TOPCLIFFE	3265	54.2039	-1.3899
LINTON ON OUSE	3266	54.0449	-1.25
LOFTUS (SAMOS)	3275	54.5629	-0.8629
FYLINGDALES	3281	54.3619	-0.6729
BRIDLINGTON MRSC	3292	54.0940	-0.1739
VALLEY	3302	53.2519	-4.5370
CAPEL CURIG SAWS	3305	53.0929	-3.9409
RHYL	3313	53.2589	-3.5090
CROSBY	3316	53.4970	-3.0559
BLACKPOOL SQUIRES GATE	3318	53.7760	-3.0369
HAWARDEN	3321	53.1739	-2.9860

Name	WMO Number	Latitude	Longitude
LEEK	3330	53.1275	-1.9799
BINGLEY SAMOS	3344	53.8110	-1.8650
WATNALL	3354	53.0050	-1.25
SCAMPTON	3373	53.3069	-0.5460
WADDINGTON	3377	53.1749	-0.5210
CRANWELL	3379	53.0309	-0.5019
LECONFIELD SAR	3382	53.8670	-0.4329
DONNA NOOK	3385	53.4729	0.1539
CONINGSBY	3391	53.0940	-0.1710
WAINFLEET	3392	53.0880	0.2739
ABERDARON	3405	52.7890	-4.7420
BALA	3409	52.9169	-3.5829
LAKE VYRNWY SAWS	3410	52.7569	-3.4639
SHAWBURY	3414	52.7939	-2.6630
WITTERING	3462	52.6110	-0.4589
HOLBEACH	3469	52.8730	0.1410
MARHAM	3482	52.6510	0.5690
WEYBOURNE	3488	52.9490	1.1269
ABERPORTH	3502	52.1389	-4.5710
TRAWSGOED	3503	52.3440	-3.9470
SENNYBRIDGE	3507	52.0629	-3.6140
SHOBDON SAWS	3520	52.2420	-2.8849
HEREFORD	3522	52.0830	-2.7999
PERSHORE	3529	52.1479	-2.0399
COLESHILL	3535	52.4799	-1.6890
CHURCH LAWFORD	3544	52.3580	-1.3300
BEDFORD	3560	52.2249	-0.4639
WATTISHAM	3590	52.1230	0.9610
MILFORD HAVEN C.B.	3604	51.7080	-5.0549
PEMBREY SANDS SAMOS	3605	51.7130	-4.3680
MUMBLES HEAD	3609	51.5649	-3.9809
FILTON	3628	51.5209	-2.5759
LITTLE RISSINGTON (ESAWS)	3647	51.8600	-1.6920

Name	WMO Number	Latitude	Longitude
BRIZE NORTON	3649	51.7579	-1.5759
BENSON	3658	51.6199	-1.0970
HIGH WYCOMBE	3660	51.6800	-0.8019
NORTHOLT	3672	51.5480	-0.4149
ANDREWSFIELD	3684	51.8959	0.4530
SHOEBURYNESS	3693	51.5540	0.8299
WALTON-ON-NAZE (CODET2)	3696	51.8540	1.2829
CHIVENOR	3707	51.0890	-4.1490
LISCOMBE	3710	51.0870	-3.6080
ST-ATHAN	3716	51.4049	-3.4400
LYNEHAM	3740	51.5031	-1.9924
LARKHILL	3743	51.2010	-1.8049
BOSCOMBE DOWN	3746	51.1609	-1.7539
MIDDLE WALLOP	3749	51.1489	-1.5700
ODIHAM	3761	51.2379	-0.9440
FARNBOROUGH	3768	51.2789	-0.7720
CHARLWOOD	3769	51.1500	-0.2333
HEATHROW	3772	51.4790	-0.4490
KENLEY	3781	51.3030	-0.0900
GRAVESEND-BROADNESS	3784	51.4640	0.3140
LANGDON BAY	3796	51.1329	1.3480
MANSTON	3797	51.3422	1.3460
SCILLY ST MARYS	3803	49.9129	-6.3010
CAMBORNE	3808	50.2179	-5.3299
CULDROSE	3809	50.0849	-5.2569
CARDINHAM	3823	50.5019	-4.6669
MOUNT BATTEN	3827	50.3540	-4.1209
EXETER AIRPORT	3839	50.7369	-3.4049
DUNKESWELL AERODROME	3840	50.8600	-3.2390
YEOVILTON	3853	51.0060	-2.6400
ISLE OF PORTLAND	3857	50.5219	-2.4539
HURN	3862	50.7789	-1.8350
ST CATHERINES PT.	3866	50.5769	-1.2970

Name	WMO Number	Latitude	Longitude
THORNEY ISLAND	3872	50.8149	-0.9229
SOLENT MRSC	3874	50.8069	-1.2079
SHOREHAM	3876	50.8359	-0.2919
HERSTMONCEUX WEST END	3882	50.8899	0.3190
CASTLEDERG	3904	54.7070	-7.5770
LOUGH FEA SAMOS	3911	54.7210	-6.8140
PORTGLENONE SAMOS	3915	54.8650	-6.4569
BALLYPATRICK FOREST	3916	55.1809	-6.1529
ALDERGROVE	3917	54.6640	-6.2239
GLENANNE	3923	54.2369	-6.5019